Extending BACOLI to solve multi-scale problems

A Thesis Submitted to the

College of Arts and Science

in Partial Fulfillment of the Requirements

for the degree of Master of Science

in the Mathematics and Statistics

University of Saskatchewan

Saskatoon

By

Elham Mirshekari

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Mathematics and Statistics
142 McLean Hall
106 Wiggins Road
University of Saskatchewan
Saskatoon, SK Canada
S7N 5E6

Abstract

The BACOLI package is a numerical software package for solving parabolic partial differential equations in one spatial dimension. It implements a B-spline collocation method for the spatial discretization of a system of partial differential equations. The resultant ordinary differential equations together with the boundary conditions form a system of differential-algebraic equations. The differential-algebraic equations are then solved using the DASSL solver. The BACOLI software package features adaptive error control in the temporal and spatial domains. The estimate of the temporal error is controlled through the DASSL solver. The estimate of the spatial error is controlled based on the difference between two solutions computed in the BACOLI software package. This difference gives an estimation of the error. If this error estimate does not meet the user-supplied tolerance, then the spatial mesh is changed.

The BACOLI software package can only solve parabolic partial differential equations that depend on spatial derivatives. In this thesis, the BACOLI software package is modified to solve a broader spectrum of problems. In fact, after some modifications, the extended BACOLI software package can solve systems of parabolic partial differential equations and time-dependent equations that do not depend on spatial derivatives. We apply this extended software package to solve the monodomain model of cardiac electrophysiology.

The monodomain model is a multi-scale mathematical model for the evolution of the electrical potential in cardiac tissue that couples the ionic currents at the cellular scale with their propagation at the tissue scale. Because of their local nature, the mathematical models of a single cell have no dependency on spatial derivatives whereas the models at the tissue level do.

The heart models considered in our numerical experiments use various cardiac cell

models. We find that solving the heart models through the extended `BACOLI` software package, in some cases, leads to a speed-up in comparison with the `Chaste` software package, which is a powerful, widely used, and well-respected software package for heart simulation.

Acknowledgements

Thank God for the wisdom and perseverance that he has been bestowed upon me during this research project, and indeed, throughout my life.

Foremost, I would like to express my sincere gratitude to my supervisor, Raymond J. Spiteri, for the continuous support of my research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and composing this thesis. I must also acknowledge Saeed Torabi Ziaratgahi and Oluwaseun Sharomi and other members of the Numerical Simulation Lab for their critiques of my work and their friendship. I would like to thank my parents, Amirnaser Mirshekari and Azam Mirmohammadian, for their emphasis on the value of education and for their unconditional support throughout my degree. A very special thanks goes to my friend and partner, Ali Satari Kasbi. He was always there cheering me up and stood by me through the good times and bad. Thanks also to Louise and Eric Braun, for their support and advice when my own parents were unavailable. I recognize that this research would not have been possible without the financial assistance of MITACS, the University of Saskatchewan Graduate Studies, the Department of Mathematics and Statistics, and National Science and Engineering Council of Canada and express my gratitude to those agencies. Last but not the least, I would like to thank my teachers: Carl de Boor, Bagher Keramati, Madjid Eshaghi Gordji, and Mahmoud Hadizadeh Yazdi.

To Ali and Soroosh

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Simulation is a powerful tool for analysing, visualizing, and predicting the behaviour of complex systems. It provides us with the possibility for new process explorations and the ability to make technical decisions. For example, the decisions can be used to decrease costs and increase the quality of processes and systems.

Mathematical simulation is an underlying branch of simulation that uses mathematics to describe real-world phenomena, survey questions about the observed phenomena, test ideas, and make predictions. The phenomena can be in engineering, physics, physiology, ecology, wildlife management, sports, chemistry, and economics Vries (2014).

Many real-world phenomena can be modelled through mathematical formulas. After the mathematical model is constructed, researchers analyse it and try to draw some conclusions. The researchers may make predictions based on their analysis. Finally, these predictions are tested to see if they are in accordance with the real-world phenomenon. The process of mathematical simulation is depicted in Figure 1.1 Vries (2014).

Mathematical simulation can be applied to many applications such as heat diffusion, plasma physics, laser pulses, seismology, quantum mechanics, chemical reactions, and biological phenomena. It has been long intertwined with one of the crucial areas in biological science, namely, the research on the propagation of the electrophysiological waves throughout the heart tissue Sundnes et al. (2006).

A specific category of the mathematical models of the heart describes how the electrophysiological waves propagate throughout the heart and cause its contractions. These models consist of systems of differential equations. The electrophysiological

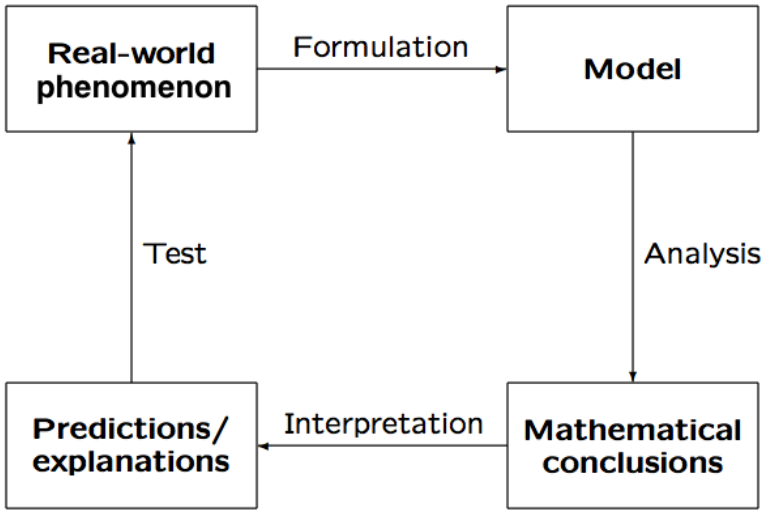

Figure 1.1: The process of mathematical simulation of a real-world phenomenon Vries (2014).

activity in the heart occurs on two different scales, the cellular and tissue scales. Activity on the cellular scale is modelled through ordinary differential equations whereas activity on the tissue scale is modelled through partial differential equations Sundnes et al. (2006).

Typically, systems of equations that mathematically describe the heart models have the following characteristics. On some time intervals, it is necessary to consider small time steps in order to guarantee numerical stability. On the other hand, there are some other time intervals on which longer time steps are desirable. Because the wave of the electrophysiological activity has a steep wavefront throughout the heart tissue, on the whole spatial domain both fine and coarse spatial meshes are desirable. There are some subintervals on which a fine spatial mesh is required. On the other hand, there are some other subintervals on which a coarse spatial mesh is desirable Whiteley (2007).

At present, finding an accurate solution for the aforementioned heart models is a significant computational challenge. In such cases, the algorithm should be efficient enough to apply appropriate time step and mesh size in the domains with high or low electrophysiological activity Trangenstein and Kim (2004).

In adaptive methods, estimating and *controlling the error* is a feature that in-

creases the reliability of the simulation. Error control means that the software package for solving the system of equations of the mathematical simulation returns an approximate solution only if the error in that approximate solution estimated to be less than a user-supplied tolerance Holder et al. (2007).

Early software packages for solving differential equations had only temporal adaptivity. Methods applied by Pormann Pormann (1999) and Quan et al. Quan et al. (1998) and software packages such as `EPDCOL` Keast and Muir (1991); Muir (2013) are of this type. In other words, these software packages estimate the error only in time and pick the time step adaptively.

More recently, in addition to temporal adaptivity, the feature of spatial adaptivity has become an integral part of an efficient error–controlling numerical software package. The number of robust software packages that employ high-order spatial and temporal discretization and enjoy error control is limited Wang et al. (2004a). These software packages are based on an algorithm that adaptively moves the mesh points within the spatial domain in order to obtain the best position such that the error in the approximate solution meets a user-supplied tolerance. The redistribution of mesh points in these algorithms is typically based on equidistribution of the spatial error estimate Wang et al. (2008). `BACOLI` Arsenault et al. (2011) is an instance of a numerical software package of this category. In fact, software packages like this compute high-order estimates of the temporal and spatial errors of an approximate solution and require these estimates to meet the user-supplied tolerance at each time step Wang et al. (2008). The problem that the `BACOLI` software package can solve is as follows:

$$\boldsymbol{u}_t(x,t) = \boldsymbol{f}(t,x,\boldsymbol{u}(x,t),\boldsymbol{u}_x(x,t),\boldsymbol{u}_{xx}(x,t)), \quad a \le x \le b, \quad t \ge t_0, \tag{1.1}$$

where $\boldsymbol{u}(x,t)$ is the unknown vector function and it is assumed that the equation is parabolic. The boundary conditions are separated and given by

$$\mathfrak{B}_a(t,\boldsymbol{u}(a,t),\boldsymbol{u}_x(a,t)) = \boldsymbol{0}, \quad \mathfrak{B}_b(t,\boldsymbol{u}(b,t),\boldsymbol{u}_x(b,t)) = \boldsymbol{0}, \quad t \ge t_0, \tag{1.2}$$

and the initial condition is given by

$$\boldsymbol{u}(x,t_0) = \boldsymbol{u}_0(x), \quad a \le x \le b, \tag{1.3}$$

where the unknown function and initial condition are m_u-dimensional vector functions.

Considering the heart simulation, there are a few algorithms that include both spatial and temporal adaptivity. The algorithm proposed by Cherry et al. Cherry et al. (2000) that is a space-time adaptive mesh refinement algorithm and the algorithm proposed by Trangenstein et al. Trangenstein and Kim (2004) that is based on a second-order operator splitting method are examples of this category that are applied for a particular group of the heart models. In Cherry et al. (2000), it has been shown that by using spatial and temporal adaptive methods, the computational effort and storage can be reduced by a factor of five. In Trangenstein and Kim (2004), speed up of the computations has been shown for a specific heart model in 1D and 2D. The total computational cost that can be reduced using an adaptive scheme is at most the ratio of the domain volume to the total volume of the regions of high electrical activity Trangenstein and Kim (2004). In Trangenstein and Kim (2004), using adaptive methods a speed up of factor seven is obtained in 1D simulations and an speed up of factor 1.7 is obtained in 2D simulations.

This thesis is on extending the `BACOLI` software package to solve multi-scale problems. The extended `BACOLI` software package can solve system of equations given by

$$\begin{aligned} u_t(x,t) &= f(t,x,\boldsymbol{v}(x,t),u(x,t),u_x(x,t),u_{xx}(x,t)),\\ \boldsymbol{v}_t(x,t) &= \boldsymbol{g}(t,x,\boldsymbol{v}(x,t),u(x,t)),\\ a \le x \le b, &\quad t \ge t_0, \end{aligned} \tag{1.4}$$

the boundary conditions are given by

$$\begin{aligned} \mathfrak{B}_a(t,\boldsymbol{v}(a,t),u(a,t),u_x(a,t)) = 0,\\ \mathfrak{B}_b(t,\boldsymbol{v}(b,t),u(b,t),u_x(b,t)) = 0, \end{aligned} \tag{1.5}$$

and the initial conditions are given by

$$\begin{aligned} u(x,t_0) &= u_0(x),\\ \boldsymbol{v}(x,t_0) &= \boldsymbol{v}_0(x), \end{aligned} \tag{1.6}$$

where $u(x,t)$ and $\boldsymbol{v}(x,t)$ are the unknown scalar and vector functions respectively. Let m_v be the size of the vector $\boldsymbol{v}(x,t)$.

Some heart models are then solved to show the performance of the extended BACOLI software package. The heart models that are considered in this study include a partial differential equation coupled with some ordinary differential equations as follows:

$$\begin{aligned} \chi C_m \frac{\partial v}{\partial t} + \chi I_{\text{ion}}(\boldsymbol{s}, v, t) &= \frac{\lambda}{1+\lambda} \nabla \cdot (\sigma_i \nabla v), \\ \frac{\partial \boldsymbol{s}}{\partial t} &= \boldsymbol{f}(\boldsymbol{s}, v, t), \end{aligned} \tag{1.7}$$

where $\boldsymbol{s}$ is a (model-dependent) vector of cellular states such as gating variables and ionic concentrations, I_{ion} is the ionic current across the membrane per unit cell membrane area that is defined in the cell model, the constant χ is the area of cell membrane per unit volume, σ_i is the conductivity, and C_m is the capacitance of the cell membrane per unit area Sundnes et al. (2006). The cell models considered in this thesis are the *FitzHugh–Nagumo*, *Luo–Rudy*, and *epicardial variant of ten Tusscher et al. (2006)* cell models. The cell models do not have any dependency on the spatial derivatives and thus cannot be solved through the BACOLI software package. However after the modifications, the extended BACOLI software package can solve these equations.

1.1 Structure of the thesis

The remainder of this thesis is organized in four chapters. Chapter 2 provides concepts of the BACOLI software package that include the numerical approaches, the structure of almost-block-diagonal matrices, and the error estimation and control schemes employed in BACOLI. Our contributions to the software package BACOLI are described in this chapter. Chapter 3 provides basic information on the physiology of the heart and gives an analysis of propagation of the electrophysiological wave on two different scales, i.e., the intracellular and extracellular scales. Chapter 4 describes the performance of the extended BACOLI software package by solving the aforementioned heart models in terms of the elapsed time required to obtain those results.

The results obtained with the extended `BACOLI` software package are compared with those obtained with `Chaste` Pitt-Francis et al. (2009). Chapter 5 gives conclusions and suggestions for future research directions.

Chapter 2

Review of BACOLI

This chapter reviews some background and basic concepts required to understand the BACOLI software package. Notions such as B-splines, almost-block-diagonal matrices, Newton's method, and differential-algebraic equations are provided to pave the way for the description of the BACOLI software package. Modifications are done to the BACOLI software package to extend it to solve multi-scale systems. These modifications are described in detail.

2.1 Collocation with B-splines

One of the methods of approximating an unknown function by piecewise polynomials includes breaking the interval into subintervals. Consider the interval $[a, b]$ divided into N_x subintervals $[x_{i-1}, x_i]$, where $i = 1, 2, \ldots, N_x$, with a partition π of the form

$$\pi : a = x_0 < x_1 < \ldots < x_{N_x} = b.$$

On each subinterval, the unknown function is approximated by a local polynomial. One can join each local polynomial to its adjacent local polynomials. The curve obtained from joining all of the local polynomials is a function $s(x) := p_{i-1}(x)$, for $x \in [x_{i-1}, x_i]$, where $i = 1, 2, \ldots, N_x$. The function $s(x)$ is called *spline* function and is a piecewise polynomial with certain properties; e.g., it can have several continuous derivatives de Boor and Inc. (1999).

The standard representation (B-representation) of a spline is as a linear sum of B-splines. A B-spline is piecewise polynomial that has minimal support with respect to its degree, continuity, and knot sequence. Considering the knot sequence

$X_1 \leq X_2 \leq \ldots \leq X_{D+M+1}$, we have

$$s(x) = \sum_{j=1}^{D} B_{j,M}(x) a_j,$$

where $B_{j,M}(x)$ is B-spline j of degree M and D is the number of the B-spline basis functions. The B-splines are generated by at most $(M+1)$ non-trivial polynomial pieces. Each $B_{j,M}(x)$ is positive on (X_j, X_{j+M+1}) and vanishes off $[X_j, X_{j+M+1}]$ de Boor (1977).

The B-spline basis functions can be defined with a divided difference scheme. Consider $\mu_i(x) := g_M(x; X_i, \ldots, X_{i+M})$, which is the divided difference M of the function $g_M(x; s) := (x - s)_+^{M-1}$, at the points $X_i, \ldots, X_{i+M}$. In this fashion, $B_{j,M}(x) = (X_{i+M} - X_i)(-1)^{M-1}\mu_i(x)$ de Boor (1977).

Other than the divided difference scheme, there is a recurrence formula for calculating the B-spline basis functions. The $B_{j,M}(x)$ can be defined recursively using the Cox–de Boor formula Dodgson (2013) as

$$B_{j,0}(x) = \begin{cases} 1, & \text{if } X_j \leq x < X_{j+1}, \\ 0, & \text{otherwise,} \end{cases}$$

$$B_{j,M}(x) = \frac{x - X_j}{X_{j+M} - X_j} B_{j,M-1}(x) + \frac{X_{j+M+1} - x}{X_{j+M+1} - X_{j+1}} B_{j+1,M-1}(x),$$

where $j = 1, 2, \ldots, D$. Figure 2.1 shows how a B-spline is made up of four pieces of the polynomials of degree three de Boor and Inc. (1999). More details about B-splines and their properties can be found in de Boor (1978).

Knots and mesh points are two different notions in B-spline studies. The knots are the points $X_i,\ i = 1, 2, \ldots, D+M+1$. Some of the knots may coincide; i.e., they may have a multiplicity greater than one. On the other hand, the mesh points are as follows. If we pick all of the knots with multiplicity one and then add to this set all of the knots with multiplicity greater than one only once, then the mesh points is constructed. An example showing the difference between these two notions is put forth shortly. Sometimes in the literature, the mesh points are referred to as the *breakpoint sequence*.

The choice of the knot sequence specifies the desired amount of continuity and smoothness at a specific knot as follows de Boor (1978);

$$(\text{number of continuity conditions at a knot}) + (\text{multiplicity of that knot}) = M + 1. \tag{2.1}$$

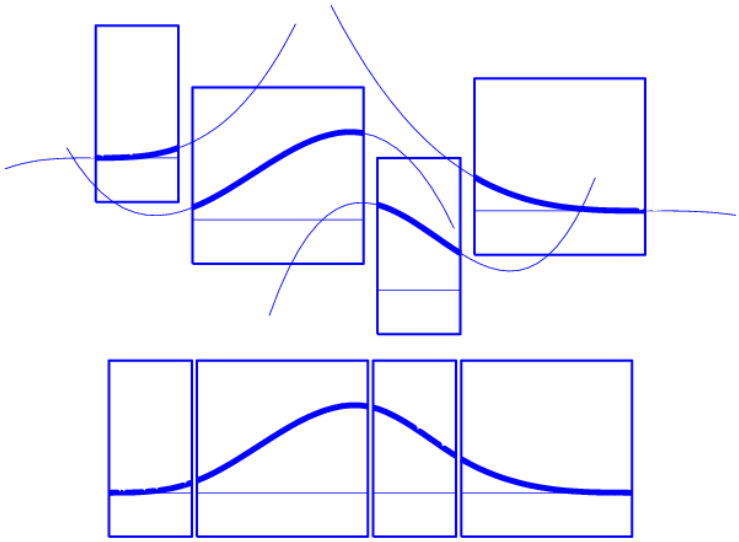

Figure 2.1: A B-spline of order four and its constituents (the four cubic polynomials) de Boor and Inc. (1999).

There are different types of knots mentioned in Dodgson (2013). The type that is considered in this thesis is of type *open uniform* and is defined as

$$\begin{cases} X_i = X_1, & \text{if } i \leq M+1, \\ X_{i+1} - X_i = \text{constant}, & M+1 \leq i < D+1, \\ X_i = X_{D+M+1}, & i \geq D+1, \end{cases}$$

where $i = 1, 2, \ldots, D+M+1$; i.e., at each boundary, the multiplicity of the knots is $(M+1)$. According to (2.1), this choice of knot sequence indicates that there is no continuity at the boundaries. An example of a B-spline defined based on this type of knot is put forth shortly.

With the choice of open uniform knots, the associated B-splines have two properties at the left boundary:

$$B_{1,M}(x_1) = 1, \tag{2.2}$$

and

$$B'_{1,M}(x_1) = -B'_{2,M}(x_1). \tag{2.3}$$

These two properties can be proven by induction and using the recursion formula of the B-splines. The proof is given in Appendix ; see also de Boor (1978); de Boor and Conte (1980). Similar properties hold for the right boundary. In the `BACOLI` software package, the knots that are considered for the B-splines are of type open uniform, and the aforementioned properties are used.

In the following example, a B-spline based on open uniform knots is presented. The difference between knots and mesh points can be seen in this example. The bold curve in Figure 2.2 shows a B-spline on $[0, 0.5]$. This B-spline is part of a spline defined on $[0, 1]$. The knot sequence is $\{0, 0, 0, 0, 0, 0, 0, 0.5, 0.5, 0.5, 0.5, 0.5, 1, 1, 1, 1, 1, 1, 1\}$, and the order of this B-spline is seven. It is of type open uniform, and the mesh point sequence is $\{0, 0.5, 1\}$. The property (2.2) can be seen in this figure; i.e., $B_{1,6}(0) = 1$. For more details on determining a B-spline, especially in the case when the multiplicity of some of the knots is greater than one, see de Boor and Conte (1980).

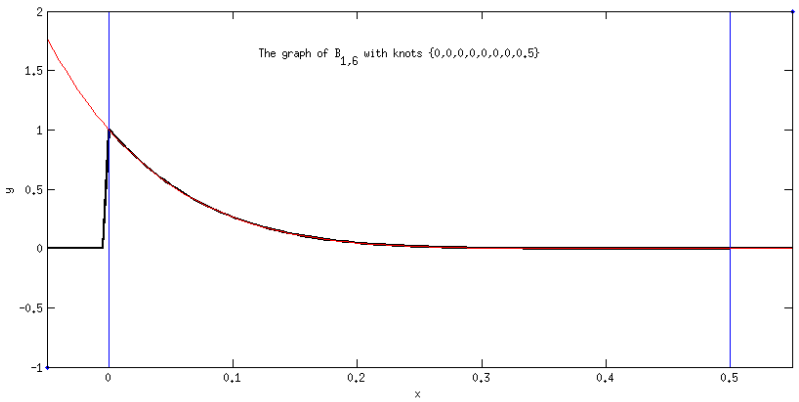

Figure 2.2: A B-spline with open uniform knots.

Consider a parabolic PDE of the form (1.1) with boundary conditions given by (1.2) and initial condition given by (1.3). In the method of spline collocation, for the solution of the problem (1.1) - (1.3), the continuous-in-time approximate solution $U(., t) \in \mathbb{P}_{M,\pi,\boldsymbol{\nu}}$, $M \geq 3$, where $\mathbb{P}_{M,\pi,\boldsymbol{\nu}}$ denotes the space of all of the piecewise

polynomial functions of degree less than or equal to M with the mesh points of π. Consider the sequence $\boldsymbol{\nu} = \{\nu_i\}_{i=1}^{N_x-1}$ whose elements denote the C^{ν_i-1}-continuity at each internal mesh point. The dimension of this space is as follows de Boor (1977):

$$D := \text{ dim } \mathbb{P}_{M,N_x,\boldsymbol{\nu}} = M + 1 + \sum_{i=1}^{N_x-1} (M + 1 - \nu_i).$$

If $\{\phi_1, \phi_2, \ldots, \phi_D\}$ is a basis for $\mathbb{P}_{M,\pi,\boldsymbol{\nu}}$, then we may write

$$U(x,t) = \sum_{k=1}^{D} \phi_k(x)\ y_k(t),$$

where the coefficients $\{y_k(t)\}_{k=1}^{D}$ are defined by requiring that U satisfies the boundary conditions, and the differential equation at $M-1$ specific points in each subinterval $[x_{i-1}, x_i]$ of π. These points, the collocation points, are defined by

$$x_{ij} = \frac{1}{2}(x_i + x_{i-1}) + \frac{1}{2}\Delta x_i \rho_j,\ i = 1, 2, \ldots, N_x,\ j = 1, 2, \ldots, M-1,$$

where $\Delta x_i = x_i - x_{i-1}$ and $\{\rho_j\}_{j=1}^{M-1}$ are the $M-1$ zeros of the Legendre polynomial of degree $M-1$ on the interval $[-1, 1]$. Because the points $\{x_{ij}\}_{j=1}^{M-1}$ are the nodes of the $(M-1)$-point Gauss quadrature rule on the interval $[x_{i-1}, x_i]$, they are commonly referred to as the Gauss points.

For the basis of $\mathbb{P}_{M,\pi,\boldsymbol{\nu}}$, the `BACOL` software package implements continuously differentiable B-splines of degree M, denoted $\{B_{j,r}\}_{j=1}^{D}$, and as a consequence the numerical method is often referred to as B-spline collocation. In this case, if the sequence $B_{1,M}, \ldots, B_{D,M}$, denote the sequence of B-splines of degree M corresponding to the knot sequence $\mathbf{X}$, where $\mathbf{X} := \{X_i\}_{i=1}^{D+M+1}$ then for a given strictly increasing sequence $\boldsymbol{x} = \{x_i\}_{i=0}^{N_x}$ as the mesh points and given non-negative integer sequence $\boldsymbol{\nu} := \{\nu_i\}_{i=1}^{N_x-1}$, this sequence of B-splines is a basis for the space $\mathbb{P}_{M,\pi,\boldsymbol{\nu}}$ de Boor (1977).

At each internal mesh point, a C^1-continuity condition is enforced; i.e.,

$$B_k(x_i) = B_{k+1}(x_i) \quad \text{and} \quad B_k'(x_i) = B_{k+1}'(x_i),$$

where $B_k(x_i)$, $i = 1, 2, \ldots, N_x - 1$, denotes B-spline function k at mesh point i.

In the case of B-splines of degree M with C^1-continuity, the set $\boldsymbol{\nu}$ is defined as follows:

$$\boldsymbol{\nu} = \{\underbrace{2, 2, \ldots, 2}_{(N_x-1) \text{ times}}\},$$

and thus the dimension of the space is

$$D = \dim \mathbb{P}_{M,N_x,\boldsymbol{\nu}} = M + 1 + \sum_{i=1}^{N_x-1} (M + 1 - 2) = N_x(M-1) + 2.$$

The exact solution of (1.1)-(1.3), $\boldsymbol{u}(x,t)$, can be approximated by a linear combination of these B-spline functions as

$$\boldsymbol{U}(x,t) = \sum_{i=1}^{D} B_i(x)\boldsymbol{y}_i(t), \tag{2.4}$$

where the functions $y_i(t)$ are the unknown coefficients. In order to determine these unknown coefficients, the `BACOL` software package requires $\boldsymbol{U}(x,t)$ to satisfy (1.1) at the collocation points of each subinterval.

The equations defining $\{y_k(t)\}_{k=1}^{M}$, the so-called collocation equations, are of the form

$$\frac{d}{dt}\boldsymbol{U}(\xi_l,t) = \boldsymbol{f}(t,\xi_l,\boldsymbol{U}(\xi_l,t),\boldsymbol{U}_x(\xi_l,t),\boldsymbol{U}_{xx}(\xi_l,t)), \tag{2.5}$$

where $l = 1 + (i-1)(M-1) + j$, $\quad i = 1, 2, \ldots, N_x$, $\quad$ and $\quad j = 1, 2, \ldots, M-1$. On the other hand, according to some properties of the B-spline basis functions, there are at most $(M+1)$ B-spline basis functions that do not vanish at each internal collocation point de Boor (1978). Generally, on the subinterval i, the sequence of B-splines $\{B_j\}_{j=k}^{l}$ does not vanish at the internal ξ_l, where $k = (i-1)(M-1)+1$ and $l = i(M-1)+2$. All of the other B-spline basis functions are zero. Taking advantage of this property,

$$\boldsymbol{U}(\xi_l,t) = \sum_{j=(i-1)(M-1)+1}^{i(M-1)+2} B_j(\xi_l)\boldsymbol{y}_j(t). \tag{2.6}$$

Inserting (2.6) into (2.5) results in the system of ODEs

$$\sum_{j=(i-1)(M-1)+1}^{i(M-1)+2} B_j(\xi_l)\boldsymbol{y}_j'(t) = \boldsymbol{f}(t,\xi_l,\boldsymbol{U}(\xi_l,t),\boldsymbol{U}_x(\xi_l,t),\boldsymbol{U}_{xx}(\xi_l,t)), \tag{2.7}$$

where $l = 1, 2, \ldots, D$. Once the system of ODEs is constructed, the discretized boundary conditions are added to this system of ODEs to form a system of initial-value DAEs.

The `BACOL` software package approximates the separated boundary conditions using (2.2) and (2.3) de Boor (1977). In other words, `BACOL` treats the boundary conditions directly, as opposed to software packages such as `PDECOL` and `EPDCOL` in which boundary conditions are differentiated Muir (2013). Differentiating the boundary conditions gives ODEs that together with ODEs obtained from the collocation method requires an ODE solver package. On the other hand, treating the boundary conditions directly gives algebraic equations that together with ODEs obtained from the collocation method requires a DAE solver package. Because differentiating the boundary conditions can cause instability to the numerical solution, it is more effective to treat the boundary conditions directly. This is an important improvement of the `BACOL` software package Muir (2013).

Utilizing the aforementioned properties, at the left boundary,

$$\begin{aligned} \boldsymbol{U}(0,t) &= B_1(0)\boldsymbol{y}_1(t), \\ \boldsymbol{U}_x(0,t) &= B_1'(0)\boldsymbol{y}_1(t) + B_2'(0)\boldsymbol{y}_2(t). \end{aligned} \tag{2.8}$$

Similarly, at the right boundary,

$$\begin{aligned} \boldsymbol{U}(1,t) &= B_D(1)\boldsymbol{y}_D(t), \\ \boldsymbol{U}_x(1,t) &= B_D'(1)\boldsymbol{y}_D(t) + B_{D-1}'(0)\boldsymbol{y}_{D-1}(t). \end{aligned} \tag{2.9}$$

Coupling the ODEs in (2.7) with the discretized boundary conditions gives an index-1 initial-value system of DAEs of the form

$$\begin{aligned} &\mathfrak{B}_a(t, \boldsymbol{U}(0,t), \boldsymbol{U}_x(0,t)) = \boldsymbol{0}, \\ &\sum_{j=(i-1)(M-1)+1}^{i(M-1)+2} B_j(\xi_l)\boldsymbol{y}_j'(t) = \boldsymbol{f}(t, \xi_l, \boldsymbol{U}(\xi_l,t), \boldsymbol{U}_x(\xi_l,t), \boldsymbol{U}_{xx}(\xi_l,t)), \\ &\mathfrak{B}_b(t, \boldsymbol{U}(1,t), \boldsymbol{U}_x(1,t)) = \boldsymbol{0}, \end{aligned} \tag{2.10}$$

where ξ_l is one of the $(M-1)$ collocation points in the subinterval i, $i = 1, 2, \ldots, N_x$. In this system, there are D equations and D unknowns. The next step is to integrate

this system of DAEs with respect to the independent variable t in order to determine the unknown coefficients $y_i(t)$.

Taking advantage of the properties of the B-splines, the matrices that are constructed in the `BACOLI` software package are sparse and, in fact, have a special structure, called almost-block-diagonal (ABD) Amodio et al. (1993). Such matrices are of the form given in Figure 2.3, where the first and last blocks correspond to the discretizations of the boundary conditions at $x = a$ and $x = b$, respectively, and the blocks $W^{(i)}$ arise from the collocation equations on the subinterval i, $i = 1, 2, \ldots, N_x$. They correspond to the discretization of the differential equation on the subintervals within the spatial domain. In this figure, the top and the bottom blocks are shown with **TOP** and **BOT** respectively, and the interior blocks are shown with $\boldsymbol{W^{(i)}}$, $i = 1, 2, \ldots, N_x$.

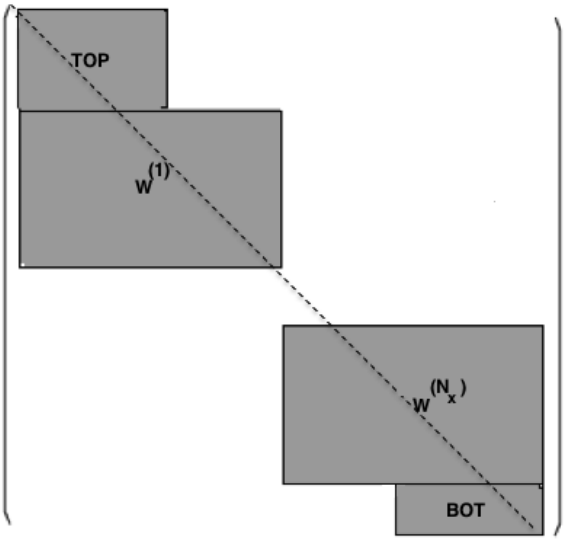

Figure 2.3: Structure of a specific almost-block-diagonal matrix with $(N_x + 2)$ blocks.

ABD matrices can be distinguished through the following four characteristics. First, the non-zero elements lie in blocks along the diagonal. Second, each diagonal element of the matrix must belong to a block. Third, any column of the ABD matrix intersects at most two blocks. Last, there are some columns that intersect two consecutive blocks, and in general, the number of these columns may vary among different pairs of the blocks.

Some works on this banded-structured matrix such as Paprzycki and Gladwell (1991) show that the algorithms involved in this type of matrix can be modified such that the computational and fill-in costs are minimized.

As can be seen in Figure 2.3, the top and the bottom blocks are not necessarily of the same size. In all of the ABD matrices that arise in the `BACOLI` software package, the overlap between a pair of consecutive blocks is a fixed number throughout the matrix. This number is equal to the number of rows in the top block plus the number of rows in the bottom block. In the `BACOLI` software package, all of the ABD matrices that arise are similar to that depicted in Figure 2.3.

Such systems arise in several commonly used numerical methods for solving two-point boundary value problems for ordinary differential equations and partial differential equations; see Amodio et al. (1993) for a comprehensive survey. ABD systems can be solved efficiently using `COLROW` Diaz et al. (1983a,b, 1988).

2.2 Newton's method to solve systems of nonlinear algebraic equations

In a system of nonlinear algebraic equations, when the number of unknowns matches the number of equations, one can attempt to solve the system through Newton's method. Systems of nonlinear algebraic equations have the form:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0, \\ f_2(x_1, x_2, \ldots, x_n) &= 0, \\ &\vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0, \end{aligned}$$

where $(x_1, x_2, \ldots, x_n)$ are the unknowns that can be written as vector $\mathbf{x}$ and $(f_1, f_2, \ldots, f_n)$ are the functions that can be written as vector $\mathbf{f}$. With vector notation, the above system can be written as

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}. \tag{2.11}$$

Newton's method converges to the root of (2.11) in an iterative process. The iterative process is

$$\mathbf{J_f}(\mathbf{x}^{(\nu)})(\mathbf{x}^{(\nu+1)} - \mathbf{x}^{(\nu)}) = -\mathbf{f}(\mathbf{x}^{(\nu)}), \quad \nu = 0, 1, \ldots, \nu^*, \tag{2.12}$$

where $\mathbf{J_f}(\mathbf{x}^{(\nu)})$ is the n by n Jacobian matrix Ascher et al. (1995) given by

$$\mathbf{J_f}(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & & \frac{\partial f_2}{\partial x_n} \\ & & & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}.$$

Starting with a sufficiently accurate initial guess $\mathbf{x}^{(0)}$, if the function $\mathbf{f}$ is smooth and its derivative can be computed, then Newton's method is convergent e.g., Ascher et al. (1995). It converges to the root of (2.11) in an iterative process. In fact, at each iteration a vector $\mathbf{x}^{(\nu)}, \nu = 0, 1, \ldots, \nu^*$, is computed until a desired accuracy is reached, i.e., $\mathbf{x}^{(\nu^*)}$ is close enough to the root. Typically $\mathbf{x}^{(0)}$ is referred to as the predictor.

2.3 Differential-algebraic equations

Consider an initial-value problem in the form of

$$\boldsymbol{G}(t, \boldsymbol{y}(t), \boldsymbol{y'}(t)) = \mathbf{0}. \tag{2.13}$$

If (2.13) can be written in the form of $\boldsymbol{y'}(t) = \boldsymbol{g}(t, \boldsymbol{y(t)})$ then (2.13) is a system of implicit ODEs; else (2.13) is called a system of differential-algebraic equations (DAEs) Brenan et al. (1996). In DAEs, there are algebraic constraints on the variables. The algebraic constraints of a system of DAEs may be semi-explicit as in

$$\boldsymbol{G_1}(\boldsymbol{y'}, \boldsymbol{y}, \boldsymbol{z}, t) = \mathbf{0}, \tag{2.14a}$$

$$\boldsymbol{G_2}(\boldsymbol{y}, \boldsymbol{z}, t) = \mathbf{0}. \tag{2.14b}$$

In this system, the constraints explicitly appear in (2.14b). For further details on the numerical solution of DAEs, see Brenan et al. (1996).

There are several applications in engineering whose mathematical models give rise to a system of DAEs. In Bauer et al. (2000), the application of DAEs in chemical processes is mentioned. In Alscher and Beyn (1998), the motion of a hydro-static skeleton is mathematically modelled and leads to a DAE system. In Riaza (2008), an optimal guidance law for spacecraft is determined using DAEs.

A system of DAEs can be transformed into a system of ODEs through taking derivatives of the constraints with respect to time t repeatedly. Numerical methods of ODEs can then be applied to solve transformed ODEs Ascher and Petzold (1998). The minimum number of the differentiations that is required to convert a system of DAEs into an ODE system is called the differential *index* of the system of DAEs Ascher and Petzold (1998).

2.4 Structure of the BACOL software package

One requires to know the concepts of the BACOL software package to understand explanations on the modifications. Because of this, in this section, the focus is on the structure of the BACOL software package. The system of equations solved through the BACOL software package is of the form (1.1) where it is assumed that the equation is parabolic. Different steps in the process of solving (1.1) through the BACOL software package together with a comparison between BACOL and BACOLI are presented. The material is largely adapted from Wang et al. (2004a).

The name BACOL is an acronym for the B-spline Adaptive COLlocation method. One of the features that makes the BACOL software package stand out among other software packages is the adaptivity in time and space. Adaptivity in time means to control an estimate of the temporal error. This feature is embedded in a routine called DASSL. Adaptivity in space means to optimize the structure of the spatial mesh at each time step, so that an estimation of the spatial error is minimized. The method for achieving the spatial adaptivity in the BACOL software package is presented shortly.

There are few software packages that apply adaptive techniques. Even fewer

of them apply high-order temporal and spatial adaptive techniques to control the error effectively. The software packages HPSIRK and HPDASSL Moore (1995), introduced by Moore, and HPNEW Moore (2001, 2014), a modified version of HPDASSL, are examples that apply high-order temporal and spatial adaptive techniques. The difference between these software packages and the BACOL software package is that in the aforementioned packages, the spatial discretization is based on a Galerkin–type discretization whereas in the BACOL software package, the discretization is based on B-splines Muir (2013). The time stepper used in HPDASSL is DASSL while in HPSIRK it is based on a singly–implicit Runge–Kutta method Muir (2013). A complete survey on the transition of a family of 1D PDE software packages from software with no spatial adaptivity to software with spatial adaptivity and error control is given in Muir (2013). As mentioned, BACOL can solve 1D system of equations as the one given in (1.1); however, an extension of the 1D B-spline Gaussian collocation algorithm used in BACOL to a 2D algorithm is proposed in Li and Muir (2013) to solve 2D PDEs.

The general scheme of a system of one-dimensional parabolic PDEs with m_u components that can be solved through the BACOL software package is of the form (1.1) where the boundary conditions are separated and given by (1.2) and the initial condition is given by (1.3). From a general perspective, the process of solving (1.1), (1.2), and (1.3) through the BACOL software package consists of the following three steps.

- First, the spatial discretization is performed using B-spline basis functions to approximate $\boldsymbol{u}(x,t)$. Inserting this approximation into (1.1) coupled with the discretized form of boundary conditions (1.2) gives a system of initial-value DAEs.

- Second, a DAE solver is applied to integrate the resulting initial-value DAEs. In the BACOL software package, the solver DASSL Petzold (1982) is used for the integration with respect to time.

- Third, the BACOL software package performs the spatial error estimate, and if necessary, it performs the spatial adaptation that involves remeshing.

The difference between BACOL and BACOLI is only in the third step. i.e., in precisely how the spatial error is to be estimated. In the rest of this section, details of these three steps are presented.

2.4.1 Spatial discretization

Without loss of generality, consider a mesh on the spatial interval $[0, 1]$ such that

$$0 = x_0 < x_1 < \ldots < x_{N_x} = 1. \tag{2.15}$$

Let $\Delta x_i = x_i - x_{i-1}$ be the length of subinterval i. In the BACOL software package, a B-spline function of degree M is allocated for each subinterval to approximate the unknown function $\boldsymbol{u}(x, t)$.

2.4.2 Time integration using the solver DASSL

In order to integrate the system (2.10), the BACOL software package employs a modified version of the solver DASSL Petzold (1982) as the time integrator. The solver DASSL is an efficient DAE solver that applies an implementation of the backward differentiation formulas (BDF) Brenan et al. (1996) for the integration of DAEs. More details on this are presented in Chapter 5 of Brenan et al. (1996). Consider the system of DAEs (2.13). The fundamental idea behind the solver DASSL is to replace the derivatives in the system of DAEs by a BDF scheme. This leads to an approximation of (2.13) of the form

$$\boldsymbol{G}\left(t_{n+1}, \boldsymbol{y}_{n+1}, \frac{\alpha}{\Delta t_{n+1}}\boldsymbol{y}_{n+1} + \boldsymbol{\beta}\right) = \boldsymbol{0}, \tag{2.16}$$

with known scalar α and vector $\boldsymbol{\beta}$, where $\Delta t_{n+1} = t_{n+1} - t_n$. One can solve the resulting equations for $\boldsymbol{y}_{n+1}$ through Newton's iteration as follows:

$$\boldsymbol{y}_{n+1}^{(\nu+1)} = \boldsymbol{y}_{n+1}^{(\nu)} - \boldsymbol{F}^{-1}\boldsymbol{G}\left(t_{n+1}, \boldsymbol{y}_{n+1}^{(\nu)}, \frac{\alpha}{\Delta t_{n+1}}\boldsymbol{y}_{n+1}^{(\nu)} + \boldsymbol{\beta}\right), \quad \nu = 0, 1, \ldots,$$

where $\boldsymbol{y}_{n+1}^{(\nu)}$ is the Newton approximation ν to $\boldsymbol{y}_{n+1}$ and $\boldsymbol{F}$ is an iteration matrix of the form

$$\boldsymbol{F} = \frac{\alpha}{\Delta t_{n+1}}\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{y'}} + \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{y}}.$$

Both of the matrices $\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{y'}}$ and $\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{y}}$ have the ABD matrix structure Wang et al. (2004b).

Without loss of generality, consider the time step t_n. At the beginning, the solver `DASSL` first computes an initial guess $\boldsymbol{y}_n^{(0)}$ for $\boldsymbol{y}_n$ by extrapolating the computed solution at the last $(k+1)$ time steps $t_{n-k-1}, \ldots, t_{n-2}, t_{n-1}$, where k is the order of the BDF scheme. In the solver `DASSL`, the order of the BDF scheme varies from one to five.

At the end of each time step, the `BACOLI` software package computes the state values and controls the error based on two tolerances, a relative tolerance and an absolute tolerance. The relative tolerance is used to measure the error relative to the size of each state. A relative tolerance of a factor of 10^{-n} means that under normal circumstances one can expect n matching digits between the exact and numerical solutions Shampine et al. (2003). The absolute tolerance is a threshold error value. When the state values drop below the absolute tolerance, they are considered insignificant Shampine et al. (2003).

Because solving the system of DAEs (2.10) leads to a Newton iteration matrix with an ABD matrix structure, it is more efficient to use a linear system solver that takes advantage of this structure. As mentioned, `COLROW` is a linear algebra software package that solves ABD linear systems efficiently. Adding the software package `COLROW` to the solver `DASSL` is the main modification in the solver `DASSL` by `BACOL`. The other modifications include improving the conditioning of the Jacobian matrices, efficient restarting of `DASSL` after a remeshing, making the initial condition consistent for `DASSL`, modifying the subroutine `DANRM` to improve its efficiency, changing the dimension of the absolute tolerance and relative tolerance to be equal to the number of equations in the system if they set to be scalars, and not allowing the order of the BDF method that is employed in the first step after a remeshing to be greater than the order used in the previous successful step. More details on these modifications are presented in Wang et al. (2004b).

The software package advances the solution in time using step sizes that are automatically selected so as to meet the user-supplied tolerances. However, the user can specify a maximum step size so that the software package avoid passing over

large intervals. Because passing over large intervals may cause numerical inaccuracy or instability in the solution, it is not desirable.

Restarting the solver `DASSL` after a remeshing: At some time steps, a given mesh may not meet the spatial error tolerance. In this case, the software package has to redefine the mesh and then redo the integration for the new mesh. The number of new mesh points, N_x^*, may be different than the number of old mesh points, N_x. In order to redo the integration for the new mesh, the solver `DASSL` interpolates the solution to the new mesh points. There are two schemes to do the interpolation of the solution to the new mesh points.

In the simpler scheme, known as a *cold start*, only the solution values from the old mesh are used to interpolate the solution to the new mesh points. Next, the integration process is performed on the new system of DAEs, where the unknowns are the B-spline coefficients on the new mesh. A cold start constrains the solver `DASSL` to apply a low-order method with a small step size at the beginning of the integration. Therefore, the level of the computational effort to perform a cold start is notably high. This gives rise to an inefficiency. Therefore, generally, this scheme is not desirable.

The second scheme is known as a *warm start*. In this scheme, not only the solution values from the latest step are used but also the solution values from as many past steps as necessary are used for the extrapolation of the solution to the new mesh. The BDF methods used in the solver `DASSL` are at most of order five. Therefore, the information of at most five time steps are required to perform a warm start. A comparison in Berzins et al. (1998) verifies that using a warm start is significantly more efficient than a cold start.

Extrapolation of the new solution is done as follows. Let $\boldsymbol{U}^*(x,t)$ be the approximate solution corresponding to the new mesh, $x_i^*, i = 0, 1, \ldots, N_x^*$. First, let the collocation points for the new mesh be $\psi_l^*, l = 1, 2, \ldots, D^*$, where $D^* = N_x^*(M-1)+2$. If the order of the BDF scheme used at the last step is k, then the new approximate solution, $\boldsymbol{U}^*(x,t)$, can be determined by the extrapolation at the new collocation

points as Wang et al. (2004a)

$$\boldsymbol{U}^*(\xi_l^*, t_{n-i}) = \boldsymbol{U}(\xi_l^*, t_{n-i}), \quad l = 1, 2, \ldots, D^*, \quad i = 0, 1, \ldots, k.$$

In the rest of this subsection, some criteria for controlling the error estimate are described. To prevent the temporal error from affecting the spatial error, the criterion that is applied for controlling the temporal error is more strict than that for the spatial error.

In the `BACOL` software package, the relative and absolute tolerances, `RTOL` and `ATOL`, are supplied by the user. The criteria for controlling the temporal error are that the absolute and relative temporal error estimates should be less than a third of the user-supplied tolerances, i.e.,

$$\text{(relative temporal error estimate)} < \frac{1}{3}\texttt{RTOL},$$

and

$$\text{(absolute temporal error estimate)} < \frac{1}{3}\texttt{ATOL}.$$

For the tolerances of the spatial error estimate, the criteria are

$$\text{(relative tolerance of spatial error estimate)} = \texttt{RTOL},$$

and

$$\text{(absolute tolerance of spatial error estimate)} = \texttt{ATOL}.$$

The solver `DASSL` controls the temporal error by enforcing these criteria.

2.4.3 Interpolation-based error estimate

In this subsection, the error estimate scheme in the `BACOL` software package is reviewed. After solving (2.10) through the solver `DASSL`, an approximate solution is available. To estimate the error in $\boldsymbol{U}(x,t)$, two approximate solutions in the space of polynomials of degree M and $(M+1)$ are constructed respectively as

$$\boldsymbol{U}(x,t) = \sum_{i=1}^{D_M} B_{M,i}(x)\boldsymbol{y}_{M,i}(t), \tag{2.17}$$

$$\tilde{\boldsymbol{U}}(x,t) = \sum_{i=1}^{D_{M+1}} B_{M+1,i}(x)\boldsymbol{y}_{M+1,i}(t), \tag{2.18}$$

where D_M and D_{M+1} are the dimensions of the space of all of the B-spline basis functions of degree less than or equal to M and $(M+1)$, respectively. For a sufficiently small step size in the spatial domain, the difference between $\boldsymbol{U}(x,t)$ and $\tilde{\boldsymbol{U}}(x,t)$ gives an asymptotic estimate of the error for the approximate solution $\boldsymbol{U}(x,t)$. The two approximate solutions $\boldsymbol{U}(x,t)$ and $\tilde{\boldsymbol{U}}(x,t)$ are determined by the solver `DASSL`. Figure 2.4 presents a schematic view of the error estimate scheme in the `BACOL` software package.

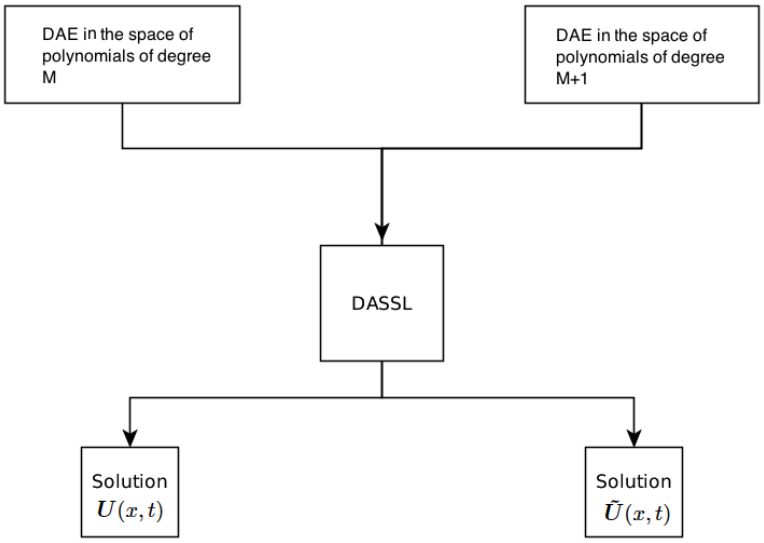

Figure 2.4: Schematic view of the error estimate scheme applied in the `BACOL` software package.

The disadvantage of the spatial error estimate scheme applied in the `BACOL` software package is that the computation of the second approximate solution, $\tilde{\boldsymbol{U}}(x,t)$, increases the cost of the computations approximately by a factor of two. Therefore, replacing this spatial error estimate scheme with a cheaper scheme that makes a satisfactory error estimate helps reduce the computational costs.

The `BACOLI` software package uses two different options to overcome the disadvantage of costly computations in the error estimate scheme of the `BACOL` software package for the estimation of the spatial error. They are through the use of a superconvergent interpolant (SCI) and a low-order interpolant (LOI). In the first option, the higher-order approximate solution, $\tilde{\boldsymbol{U}}(x,t)$ in (2.18), is replaced with an interpolant. In the second option, the lower-order approximate solution, $\boldsymbol{U}(x,t)$ in (2.17),

is replaced with an interpolant Arsenault et al. (2011). In the next subsection, more details on these two options are put forth.

2.4.4 Overview of the SCI and LOI options

In this subsection, the two options by which the `BACOLI` software package estimates the spatial error are presented. First, an overview of the SCI option is put forth. This is followed by an overview of the LOI option. The material in this subsection is mostly adapted from Arsenault et al. (2011).

In the SCI option Arsenault et al. (2009), at any time step, only the lower-order approximate solution is computed through the solver `DASSL`. Based on this solution, which is of degree M, an interpolant of degree $(M+1)$ can be constructed using some super-convergent values. After that, the interpolant can replace $\tilde{\boldsymbol{U}}(x,t)$ in (2.18). A schematic view of this option can be seen in Figure 2.5. In this figure, the empty box indicates the difference between the SCI option and the error estimate scheme applied in the `BACOL` software package. In other words, it indicates that there is no DAE in the space of polynomials of degree $M+1$ to be passed to `DASSL`.

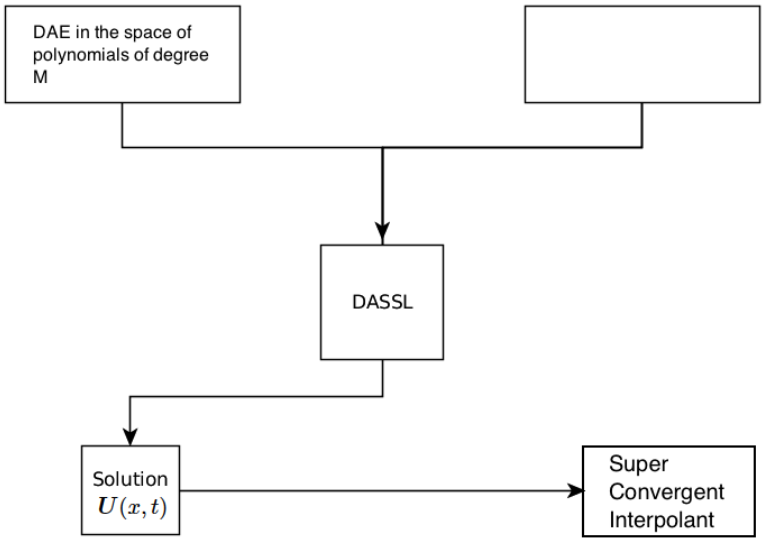

Figure 2.5: Schematic view of the super-convergent interpolant (SCI) option in the `BACOLI` software package.

In Arsenault et al. (2011), Arsenault et al. tentatively present some results of the collocation error. These results are based on the piecewise polynomials of degree M

that are associated with $(M-1)$ collocation points per subinterval. They show that first, at the mesh points, derivative j of the collocation error has order $2(M-1)$ i.e.,

$$|\boldsymbol{u}^{(j)}(x_i,t) - \boldsymbol{U}^{(j)}(x_i,t)| = \mathcal{O}(\Delta^{2(M-1)}), \qquad i = 0, 1, \ldots, N_x + 1,$$

where the superscript, j, indicates derivative j of the function, $\Delta x_i = x_{i+1} - x_i$, and $\Delta = \max(\Delta x_i),\ i = 0, 1, \ldots, N_x$.

Second, at non-mesh points, derivative j of the collocation error satisfies

$$\begin{aligned}\boldsymbol{u}^{(j)}(x_i,t) - \boldsymbol{U}^{(j)}(x_i,t) &= \frac{1}{(M-1)!}\Delta x_i^{M+1-j}\boldsymbol{u}^{(M+1)}(x_i,t)P_{M-1}^{(j)}\left(\frac{x-x_i}{\Delta x_i}\right) + \mathcal{O}(\Delta x_i^{M+2-j}) \\ &\quad + \mathcal{O}(\Delta^{2(M-1)}) \\ &= \mathcal{O}(\Delta^{M+1-j}),\end{aligned}$$

where $j = 0, 1, \ldots, M$, $i = 0, 1, \ldots, N_x$, and

$$P_{M-1}(\psi) = \int_0^{\psi} (t-\psi) \prod_{l=1}^{M-1} (t-\rho_l) dt,$$

where the points ρ_l are the images of the $(M-1)$ Gauss points in $[0, 1]$. Accordingly, the super-convergence property at the mesh points holds if the order of the collocation error at the mesh points, $2(M-1)$, is greater than the order of the collocation error at the non-mesh points, $(M+1)$; i.e., $M > 3$.

On the other hand, the roots of the polynomial $P_{M-1}(\psi)$ can lead to a higher accuracy in the collocation solution if $2(M-1) \geq M+2$; i.e., if $M \geq 4$. The roots of the first derivative of the polynomial $P_{M-1}(\psi)$ can also lead to a higher accuracy in the first derivative of the collocation solution if $2(M-1) \geq M+1$; i.e., if $M \geq 3$.

In a nutshell, the super-convergent points mentioned above are composed of the mesh points at which both of the collocation solution and its derivative offer higher accuracy. They consist of the roots of $P_{M-1}(\psi)$, at which only the collocation solution offers higher accuracy, and the roots of $P'_{M-1}(\psi)$, at which only the first derivative of the collocation solution offers higher accuracy.

Let us suppose that the order of the accuracy of the solution and the order of the accuracy of the interpolation are $\mathcal{O}(\Delta^{p_1})$ and $\mathcal{O}(\Delta^{q_1})$ respectively. For the interpolation error not to corrupt the accuracy of the solution, one has to impose

the condition $q_1 > p_1$. When the degree of the piecewise polynomials is M, the order is $(M+1)$, and the order of the accuracy of the solution at the non-mesh super-convergent points is $(M+2)$. Thus, the order of the interpolation error, q_1, should be greater than the order of the accuracy of the solution, $(M+2)$; i.e., $q_1 > M+2$. In other words, the minimum amount that q_1 can take is $(M+3)$. On the other hand, in order to have an interpolant of order $(M+3)$ one requires at least $(M+3)$, solution values per subinterval.

To impose C^1-continuity, the four super-convergent values of the solution and its first derivative at the endpoints of each subinterval are taken into account in the interpolation process. Therefore, one must choose the remaining $(M+3)-4 = M-1$ super-convergent points for the super-convergent interpolation. The interpolant that is used in the `BACOLI` software package is of type Hermite–Birkhoff Arsenault et al. (2011) and is explained shortly. Because of the choice of this interpolant and its existence issues, these values cannot be chosen from the super-convergent derivative values of that subinterval. An alternative is to choose all of the $(M-3)$ super-convergent solution values within that subinterval. The remaining two values are chosen from the closest available super-convergent solution values from the left and right adjacent subintervals. However, for the leftmost and rightmost subintervals, one can choose the two closest super-convergent solution values available in its single adjacent subinterval.

Once the interpolation points are chosen, it is time to interpolate. On each subinterval $[x_i, x_{i+1}]$ let $s_1 = x_i$ and $s_2 = x_{i+1}$. Also let $\bar{s}_j,\ j = 1, 2, \ldots, (M-1)$, be the non-mesh super-convergent points. Then for the SCI option, on the given subinterval and at time t, $\tilde{\boldsymbol{U}}(x,t)$ in (2.18) is replaced by Hermite–Birkhoff SCI as follows;

$$\tilde{\boldsymbol{U}}(x,t) = \sum_{j=1}^{2} H_j(x)\boldsymbol{U}(s_j,t) + h\sum_{j=1}^{2} \bar{H}_j(x)\boldsymbol{U}'(s_j,t) + \sum_{j=1}^{(M-1)} G_j(x)\boldsymbol{U}(\bar{s}_j,t), \qquad (2.19)$$

where $x \in [x_i, x_{i+1}]$ and $\boldsymbol{U}(x,t)$ in (2.17) is known after the time integration by the

solver DASSL; $H_j(x)$, $\bar{H}_j(x)$, and $G_j(x)$ are given by

$$H_j(x) = (1-(x-s_j)\gamma_j)\frac{\xi_j^2(x)\phi(x)}{\xi_j^2(s_j)\phi(s_j)},$$
$$\bar{H}_j(x) = (x-s_j)\frac{\xi_j^2(x)\phi(x)}{\xi_j^2(s_j)\phi(s_j)},$$
$$G_j(x) = \frac{\xi^2(x)\phi_j(x)}{\xi^2(\bar{s}_j)\phi_j(\bar{s}_j)},$$

where

$$\phi(x) = \prod_{r=1}^{M-1}(x-\bar{s}_r), \quad \phi_j(x) = \prod_{\substack{r=1\\ r\neq j}}^{M-1}(x-\bar{s}_r),$$
$$\xi(x) = \prod_{r=1}^{2}(x-s_r), \quad \xi_j(x) = \prod_{\substack{r=1\\ r\neq j}}^{2}(x-s_r),$$

and

$$\gamma_j = \sum_{i=1}^{M-1}\frac{1}{s_j-\bar{s}_i} + 2\sum_{\substack{i=1\\ i\neq j}}^{2}\frac{1}{s_j-s_i}.$$

The description of the LOI option Arsenault et al. (2012) is as follows. At any time step, only the solution $\tilde{\boldsymbol{U}}(x,t)$ in (2.18) is computed through the solver DASSL. Based on this solution, a lower-order interpolant is constructed. The error estimate obtained from this option asymptotically approaches to the error estimate for the collocation solution with degree M. In Arsenault et al. (2011), Arsenault et al. tentatively determine the error expansion for the degree-M collocation solution, $\boldsymbol{U}(x,t)$, at the time t and on the subinterval $[x_i, x_{i+1}]$ as

$$\boldsymbol{u}(x,t)-\boldsymbol{U}(x,t) = \frac{1}{(M-1)!}\Delta x_i^{M+1}\boldsymbol{u}^{(M+1)}(x_i,t)P_{M-1}\left(\frac{x-x_i}{\Delta x_i}\right)+\mathcal{O}(\Delta x_i^{M+2})+\mathcal{O}(\Delta^{2(M-1)}),$$

where

$$P_{M-1}(\psi) = \int_0^{\psi}(t-\psi)\prod_{r=1}^{M-1}(t-\rho_r)dt,$$

$(M-1)$ is the number of collocation points per subinterval, Δ is the maximum subinterval size, and the points ρ_r are the $(M-1)$ Gauss points mapped on $[0,1]$.

For the interpolation error, which is of order $\mathcal{O}(\Delta x^{q_1})$, not to corrupt the accuracy of the solution, which is of order $\mathcal{O}(\Delta x^{p_1})$, one has to impose the condition $q_1 > p_1$.

In this case, this condition leads to the condition that q_1 should be at least $(M+1)$. Thus, for the interpolation, one has to choose $(M+1)$ points per subinterval.

Accordingly, in the LOI option, the number of required interpolation points is two fewer than that for the SCI option. It appears that the super-convergent values should be similar to those for the SCI option except for the two interpolation points outside of the subinterval. Thus, the values used for the interpolation in the LOI option per subinterval include the solution and derivative values at the endpoints of each subinterval plus the solution values in that subinterval.

Once the interpolation points are specified, it is time to interpolate. The interpolant is again Hermite–Birkhoff which mentioned previously in (2.19). Let $s_1 = x_i$, $s_2 = x_{i+1}$, and $\bar{s}_j,\ j = 1, 2, \ldots, M-3$, be the non-mesh super-convergent points internal to the subinterval $[x_i, x_{i+1}]$. Based on the higher-order collocation solution, $\tilde{\boldsymbol{U}}(x,t)$, that is obtained through integration by the solver `DASSL`, the Hermite–Birkhoff LOI scheme is as follows;

$$\boldsymbol{U}(x,t) = \sum_{j=1}^{2} H_j(x)\tilde{\boldsymbol{U}}(s_j,t) + \Delta x_i \sum_{j=1}^{2} \bar{H}_j(x)\tilde{\boldsymbol{U}}'(s_j,t) + \sum_{j=1}^{M-3} G_j(x)\tilde{\boldsymbol{U}}(\bar{s}_j,t),$$

where $x \in [x_i, x_{i+1}]$ and $\Delta x_i = x_{i+1} - x_i$. The basis functions $H_j(x)$, $\bar{H}_j(x)$, and $G_j(x)$ are the same as those defined for the Hermite–Birkhoff SCI.

Finally, in order to estimate the error through this scheme, the solution $\boldsymbol{U}(x,t)$ in (2.17) is replaced by the Hermite–Birkhoff LOI. A schematic view of this option can be seen in Figure 2.6. In this figure, the empty box indicates the difference between the LOI option and the error estimate scheme applied in the `BACOL` software package. In other words, it indicates that there is no DAE in the space of polynomials of degree M to be passed to `DASSL`.

Because the SCI scheme generates a numerical solution and an error estimate for that solution, one can consider this option as a standard error control mode. If the software package runs with the LOI error estimate, that error estimate is actually for a collocation solution that is one order lower than the collocation solution that is returned by the `BACOLI` software package. In the case of LOI option, one can consider that the `BACOLI` software package is running in local extrapolation mode.

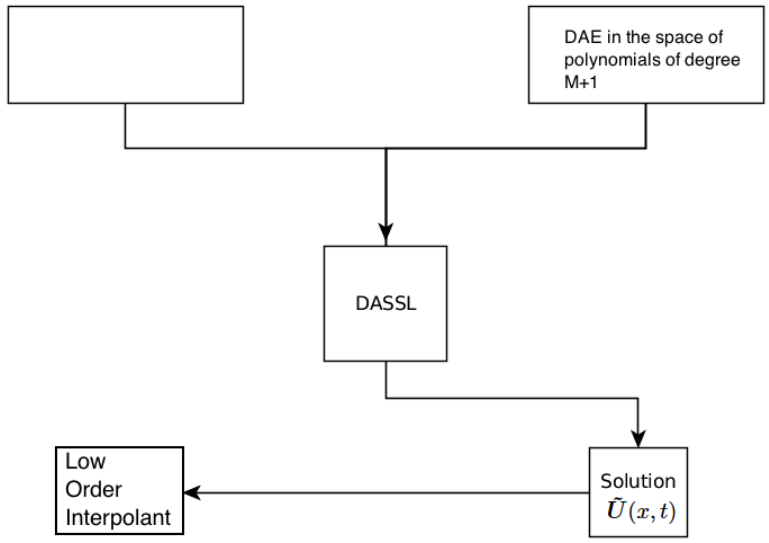

Figure 2.6: Schematic view of the low-order interpolant (LOI) option in the `BACOLI` software package.

2.5 Mesh refinement

The strategy for the mesh refinement in the `BACOLI` software package is similar to the one in the `BACOL` software package. It is based on two phases. First, the number of mesh points is determined. Second, the location of the new mesh points is determined. In the rest of this subsection, the assumption is that the software package is `BACOLI` and the error estimate scheme is SCI.

For estimating the spatial error, let the variables without bars denote the approximate solution obtained from the solver `DASSL` and those with bars denote the interpolant. For example, ξ_l and $\boldsymbol{U}(x,t)$ are the collocation point l and the approximate solution respectively that are associated with the solution obtained from the solver `DASSL`. On the other hand, $\bar{\xi}_l$ and $\bar{\boldsymbol{U}}(x,t)$ are the collocation point l and the approximate solution respectively that are associated with the interpolant.

The $\mathfrak{L}^2$-norm of the error estimate on $[x_{i-1}, x_i]$ is

$$\|\boldsymbol{U}(x,t) - \bar{\boldsymbol{U}}(x,t)\|_2 = \sqrt{\int_{x_{i-1}}^{x_i} (\boldsymbol{U}(x,t) - \bar{\boldsymbol{U}}(x,t))^2 dx},$$

and the $\mathfrak{L}^2$-norm of the exact error is given by

$$\|\boldsymbol{u}(x,t) - \boldsymbol{U}(x,t)\|_2 = \sqrt{\int_{x_{i-1}}^{x_i} (\boldsymbol{u}(x,t) - \boldsymbol{U}(x,t))^2 dx}.$$

A quasi-uniform mesh is the type of mesh that for some constant K, $\Delta x_i \leq K\Delta_{\min}$, where Δx_i is the step size and $\Delta_{\min}$ is the minimum of the step sizes Wang et al. (2004a). In the boundary-value ODEs solved with a numerical method of order $(M+1)$ and quasi-uniform mesh,

$$\|\boldsymbol{u}(x,t) - \boldsymbol{U}(x,t)\| = \mathcal{O}(\Delta^{M+1}) \text{ as } \Delta \to 0. \tag{2.20}$$

A similar property is given shortly for the error estimate used in the `BACOLI` software package.

For a nice definition of the error estimate used in the `BACOLI` software package, let us consider the general case of m_u PDEs. As mentioned, `RTOL` and `ATOL` are the relative and absolute temporal tolerances, respectively. In the case of a system of m_u equations, they could be vectors of dimension m_u. Let $\texttt{ATOL}_j$ and $\texttt{RTOL}_j$ be the tolerances corresponding to the component j of the PDE system. The `BACOLI` software package applies two types of error estimates, both of which are normalized. The logic behind using two different types of error estimates is discussed shortly. In fact, the `BACOLI` software package appropriately chooses one of them in different situations.

The first type of normalized error estimate, $E_j(t)$, is defined for each component of the PDE as follows:

$$E_j(t) = \sqrt{\int_0^1 \left(\frac{U_j(x,t) - \bar{U}_j(x,t)}{\texttt{ATOL}_j + \texttt{RTOL}_j |U_j(x,t)|} \right)^2 dx}, \quad j = 1, 2, \ldots, m_u. \tag{2.21}$$

The second type, $\hat{E}_i(t)$, is defined on each subinterval for all of the components of the PDE system as follows:

$$\hat{E}_i(t) = \sqrt{\sum_{j=1}^{m_u} \int_{x_{i-1}}^{x_i} \left(\frac{U_j(x,t) - \bar{U}_j(x,t)}{\texttt{ATOL}_j + \texttt{RTOL}_j |U_j(x,t)|} \right)^2 dx}, \quad i = 1, 2, \ldots, N_x. \tag{2.22}$$

In both of the cases, t is the end of the time step on which the solver `DASSL` is working.

By experimental tests, Wang et al. Wang et al. (2004a) show that

$$E \equiv \max_{1 \leq j \leq m_u} E_j(t) \propto \mathcal{O}(\Delta)^{M+1}, \tag{2.23}$$

and this property is used in the mesh refinement process.

There are two spatial error estimates in the BACOLI software package, each of which is used in different situations. The definition of these two error estimates is as follows. The first spatial error test is based on the values $\hat{E}_i$ from the equation (2.22) and E_j from the equation (2.21). For the first spatial error estimate, the BACOLI software package first computes the parameters r_1 and r_2 as follows.

$$\begin{cases} r_1 = \max\limits_{1 \le i \le N_x} (\hat{E}_i)^{1/(M+1)}, \\ r_2 = \dfrac{1}{N_x} \sum\limits_{i=1}^{N_x} (\hat{E}_i)^{1/(M+1)}. \end{cases}$$

Having the parameters r_1 and r_2, one can have a criterion to determine if a given mesh is well-distributed. For example, the BACOLI software package considers the mesh well-distributed if

$$\frac{r_1}{r_2} \le 2. \tag{2.24}$$

There is another criterion that is used to determine if the user-supplied tolerance is met. In fact, if

$$E \equiv \max_{1 \le s \le m_u} E_s \le 1, \tag{2.25}$$

it means that the user-supplied tolerance is met.

The first spatial error estimate can be stated as the two aforementioned criteria. In other words, the two criteria (2.24) and (2.25) together make the first spatial error estimate. If one of them does not hold, the first spatial error estimate is considered to not meet the tolerance. Therefore, a remeshing is required, and an adaptive strategy is applied.

The second spatial error estimate is to some extent similar to the first spatial error estimate. If $\hat{E}_i$ from (2.22) satisfies (2.24), and E from (2.25) satisfies

$$0.1 < E < 0.4, \tag{2.26}$$

then the second spatial error estimate holds. In fact, after a successful time step by the solver DASSL, the second spatial error estimate is applied to determine if the current step is accepted.

The second spatial error estimate is stricter than the first spatial error estimate. The BACOLI software package uses the second spatial error estimate for the first step after any remeshing or for the first step of the time integration. If a mesh distribution passes the second spatial error estimate, then the BACOLI software package switches back the criterion to the first spatial error estimate for the future time steps. The BACOLI software package uses the first spatial error estimate until it reaches a time step where this test fails. This failure means that a redistribution of the mesh points is necessary. The BACOLI software package then discards the latest past time step and switches to the second spatial error test after the redistribution. In this way, the BACOLI software package switches between these two criteria. The numerical experiments in Wang et al. (2004a) show the efficiency of this double-criteria method.

The strategy of determining the number of the new mesh points is obtained based on some numerical experiments by Wang et al. Wang et al. (2004a). Let N_x be the number of mesh points at the current time step and suppose that the first spatial error test indicates that a remeshing is performed. If it is the initial remeshing at this time step, then let $N_x^* = N_x$. Because when the first spatial error test fails, usually by redistributing the same number of mesh points the solver DASSL can do the integration successfully. A warm start should then be applied.

If the remeshing is to be done for the second, third, or fourth time, then N_x^* is determined by a strategy that is mentioned shortly. Again a warm start is applied.

If the remeshing is to be done for the fifth time, then the BACOLI software package sets $N_x^* = N_x$ and tries a cold start.

If the remeshing is to be done for the sixth to twentieth time, then the BACOLI software package determines N_x^* and again tries a cold start. Finally, if all of the twenty remeshings fail, then the BACOLI software package raises an error.

The strategy for computing the number of new mesh points, N_x^*, is as follows. Assume that a remeshing is done with $N_x^* = N_x$ but the second spatial error estimate does not hold. Numerical experiments justify that in most of the cases because (2.26) does not hold, the second spatial error estimate is failed. From (2.23), it can be

obtained that

$$E = \mathcal{O}(\Delta^{M+1}).$$

Experimental results show that the number of new mesh points is optimized if the normalized error estimate after the remeshing, E^*, be equal to 0.2. In other words,

$$0.2 = E^* \propto (N_x^*)^{-(M+1)}. \tag{2.27}$$

Dividing (2.23) by (2.27) gives rise to

$$\frac{E}{0.2} \propto \left(\frac{N_x^*}{N_x}\right)^{M+1}.$$

After $\frac{E}{0.2} = (\frac{N_x^*}{N_x})^{M+1}$ is set, N_x^* is determined as follows Wang et al. (2004a):

$$N_x^* = N_x \left(\frac{E}{0.2}\right)^{1/(M+1)}.$$

If (2.26) holds, then N_x^* is taken to be the same as N_x. If $E \geq 0.4$, then the resulting N_x^* is greater than N_x, meaning that a greater number of subintervals is required. If $E \leq 0.1$, then the resulting N_x^* is less than N_x, meaning that a lower number of subintervals is required.

After determining the number of new mesh points, the `BACOLI` software package uses an equidistribution principle for generating the new mesh structure. In other words, the new mesh points, $\{x_i^*\}_{i=1}^{N_x^*}$, should satisfy

$$\sqrt{\sum_{j=1}^{m_u} \int_{x_{i-1}^*}^{x_i^*} \left(\frac{U_j - \bar{U}_j}{\texttt{ATOL}_j + \texttt{RTOL}_j |U_j|}\right)^2 dx} = \left(\frac{\sum_{i=1}^{N_x} \hat{E}_i^{1/(M+1)}}{N_x^*}\right)^{M+1},$$

where the right-hand side (RHS) is a constant. In fact, the normalized error estimate for all of the new subintervals are equal.

2.5.1 Algorithm overview

A flowchart of the `BACOLI` software package is presented in Figures 2.7, 2.8, 2.9, 2.10, and 2.11. The general scheme of the `BACOLI` software package is presented in Figure 2.7, in which there are seven different labels.

The last three labels, i.e., labels 500, 600, and 700, are different terminals as follows. First, if the software package gets a proper input and the `BACOLI` software package solves the problem successfully, then the software package ends up at the terminal 500. Some files including the outputs are produced accordingly. Second, if the `BACOLI` software package cannot solve the problem, then the software package ends up at the terminal 600. In this case, a message is printed to indicate the error. Finally, if there is an invalid input, then the software package ends up at the terminal 700. A message is printed to indicate the invalid input accordingly.

The four first labels, i.e., labels 100, 200, 300, and 400, serve the following purposes. Label 100 checks if the input is valid and it also initializes the work space at the first call to the `BACOLI` software package or after any remeshing. The schematic view of this label is presented in Figure 2.8.

Label 200 checks if it is the first call to the problem or not and based on this, it continues the integration process. The schematic view of this label is presented in Figure 2.9.

Label 300 performs the initialization tasks after any remeshing. The schematic view of this label is presented in Figure 2.10.

Label 400 performs the time integration. Because the time integration is performed in this label, it is one of the most important parts of the algorithm. The solver `DASSL` is called from this label. The schematic view of this label is presented in Figure 2.11.

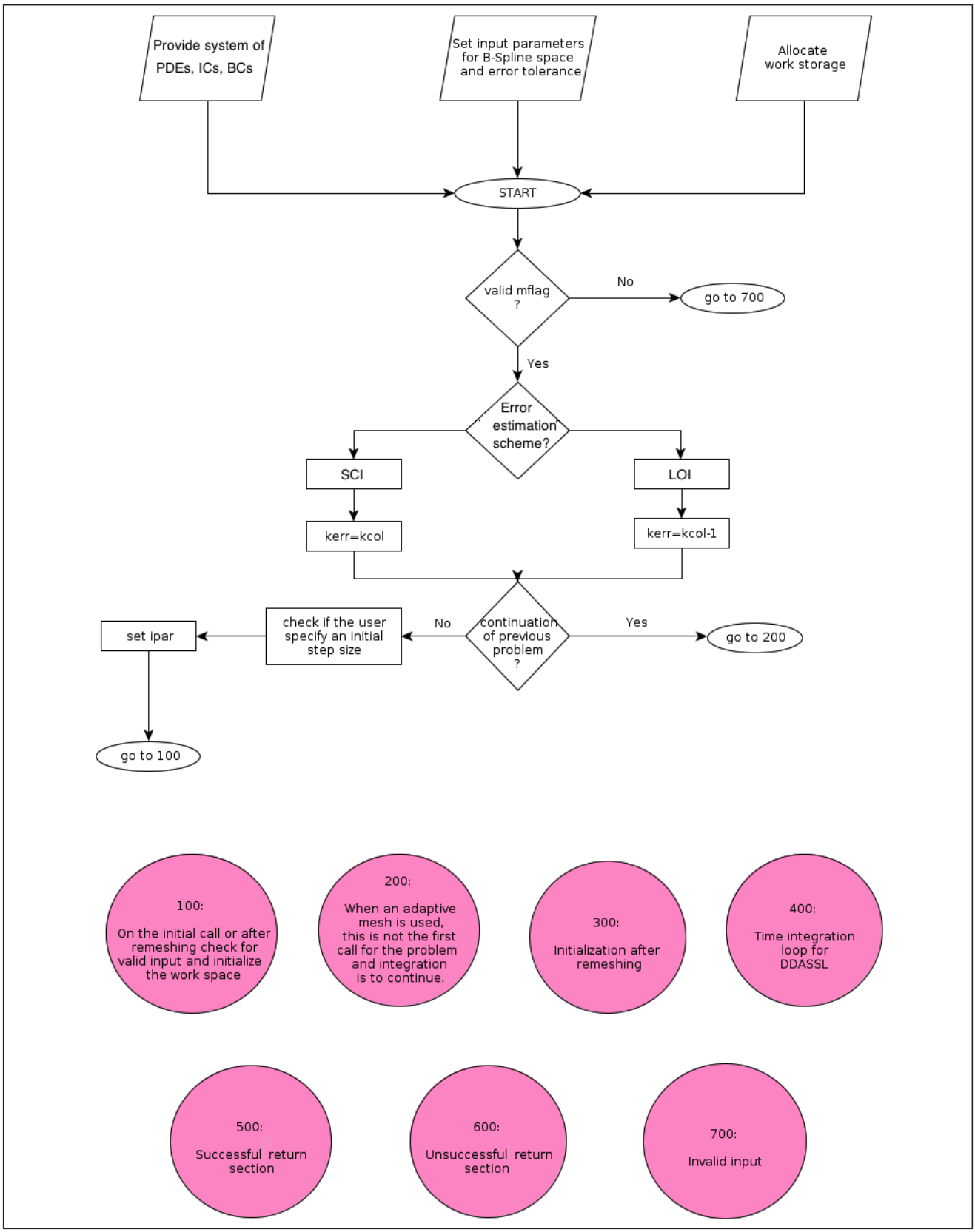

Figure 2.7: The general scheme of `BACOLI`.

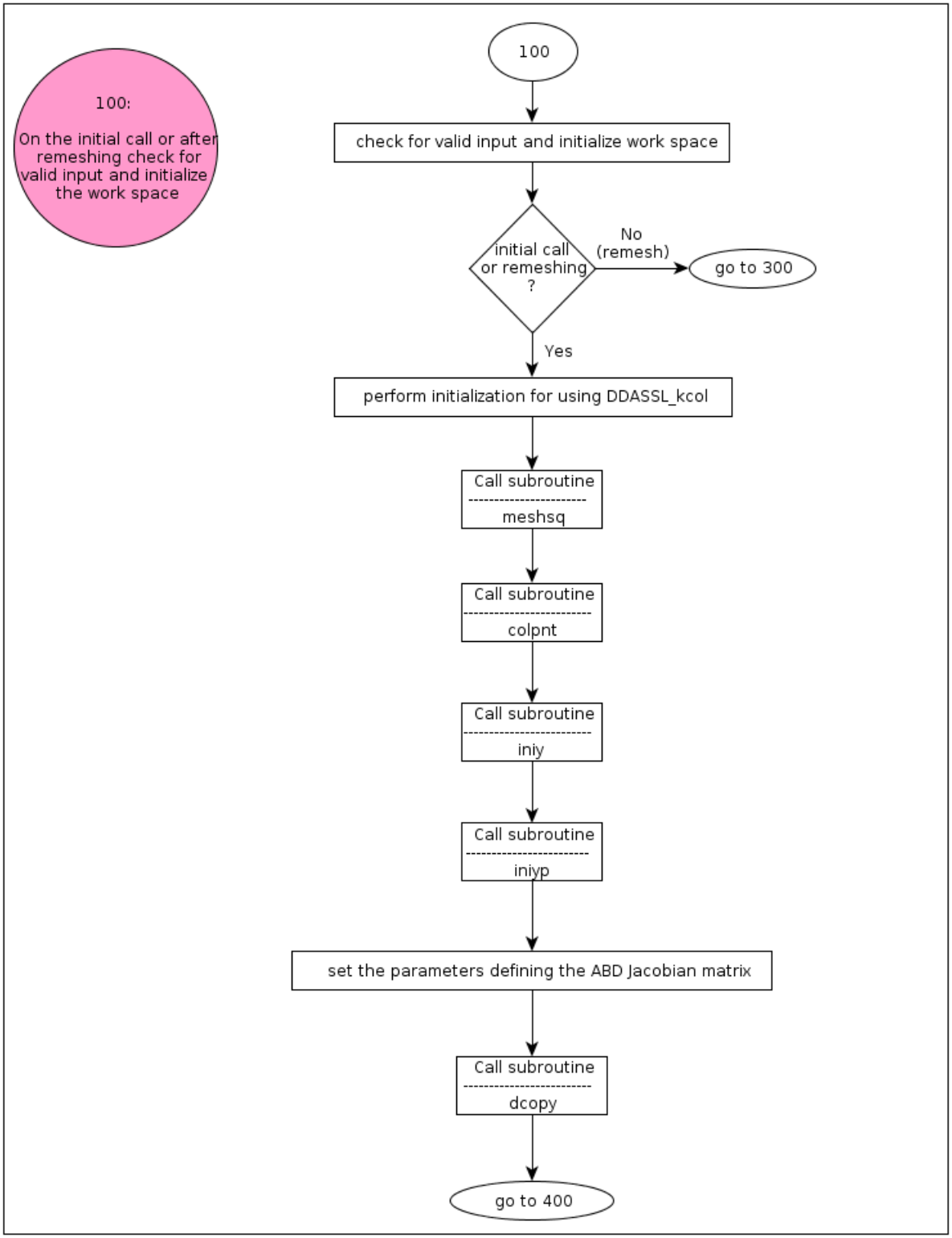

Figure 2.8: Label 100 of the `BACOLI` algorithm.

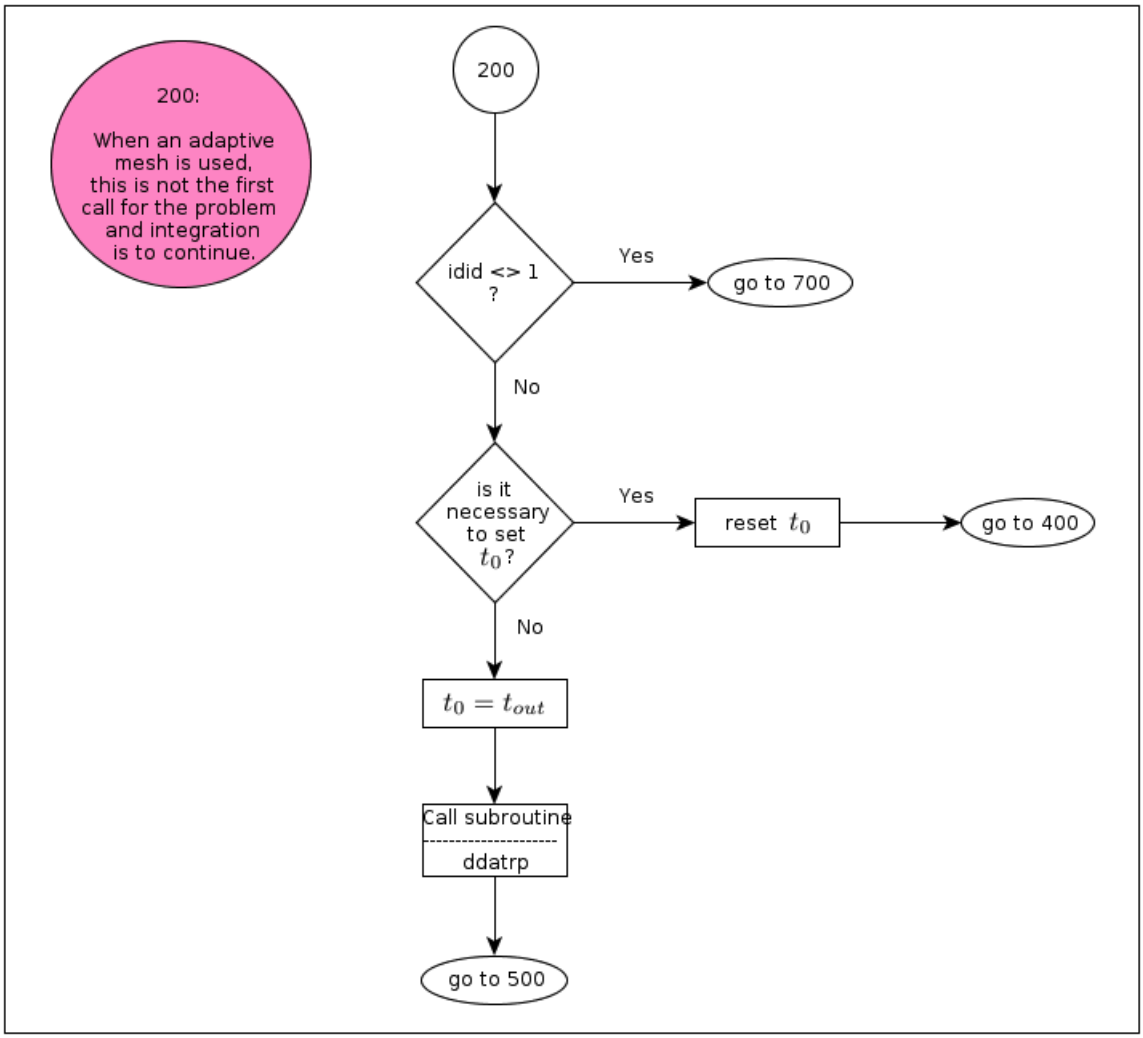

Figure 2.9: Label 200 of the `BACOLI` algorithm.

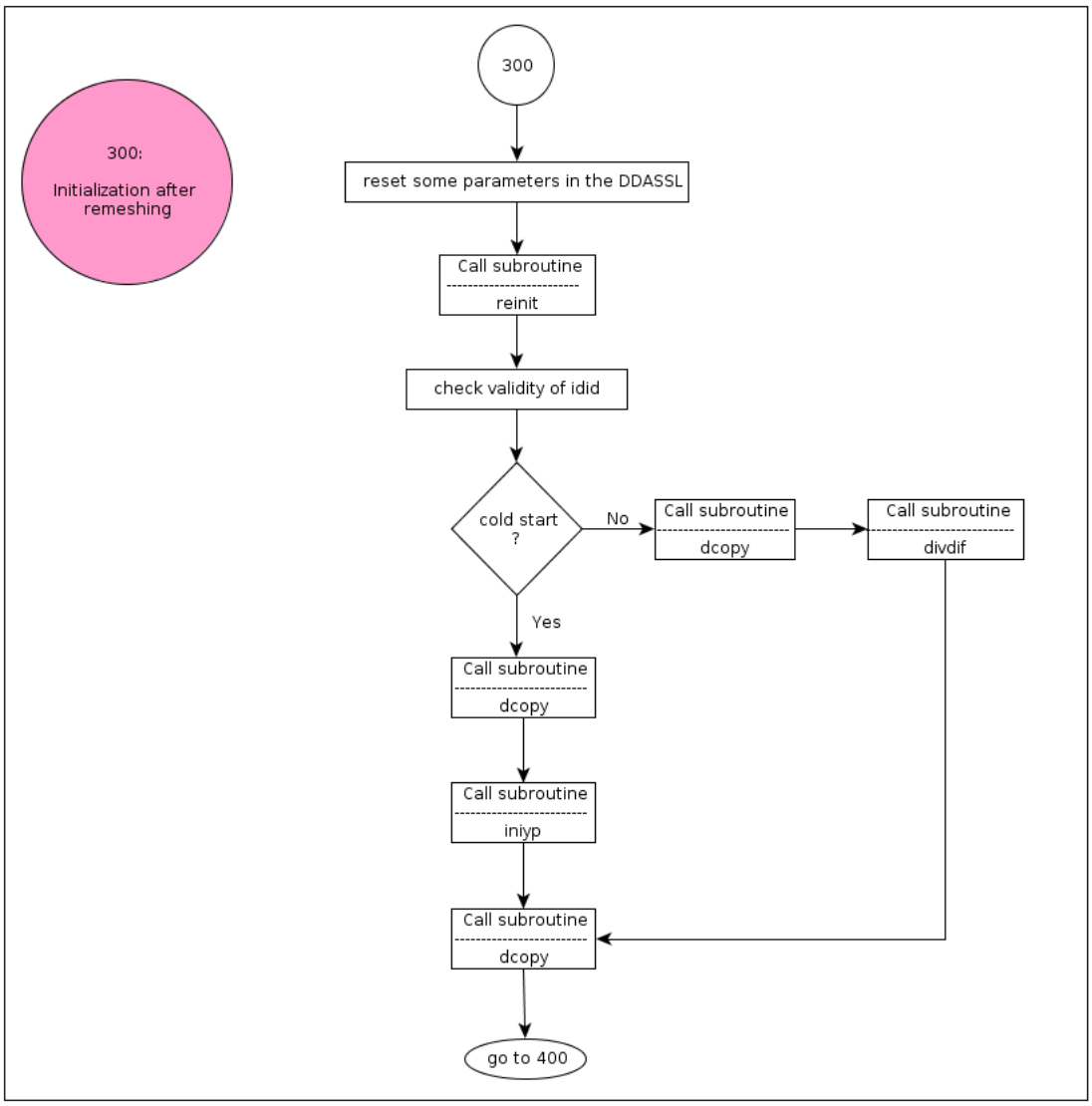

Figure 2.10: Label 300 of the `BACOLI` algorithm.

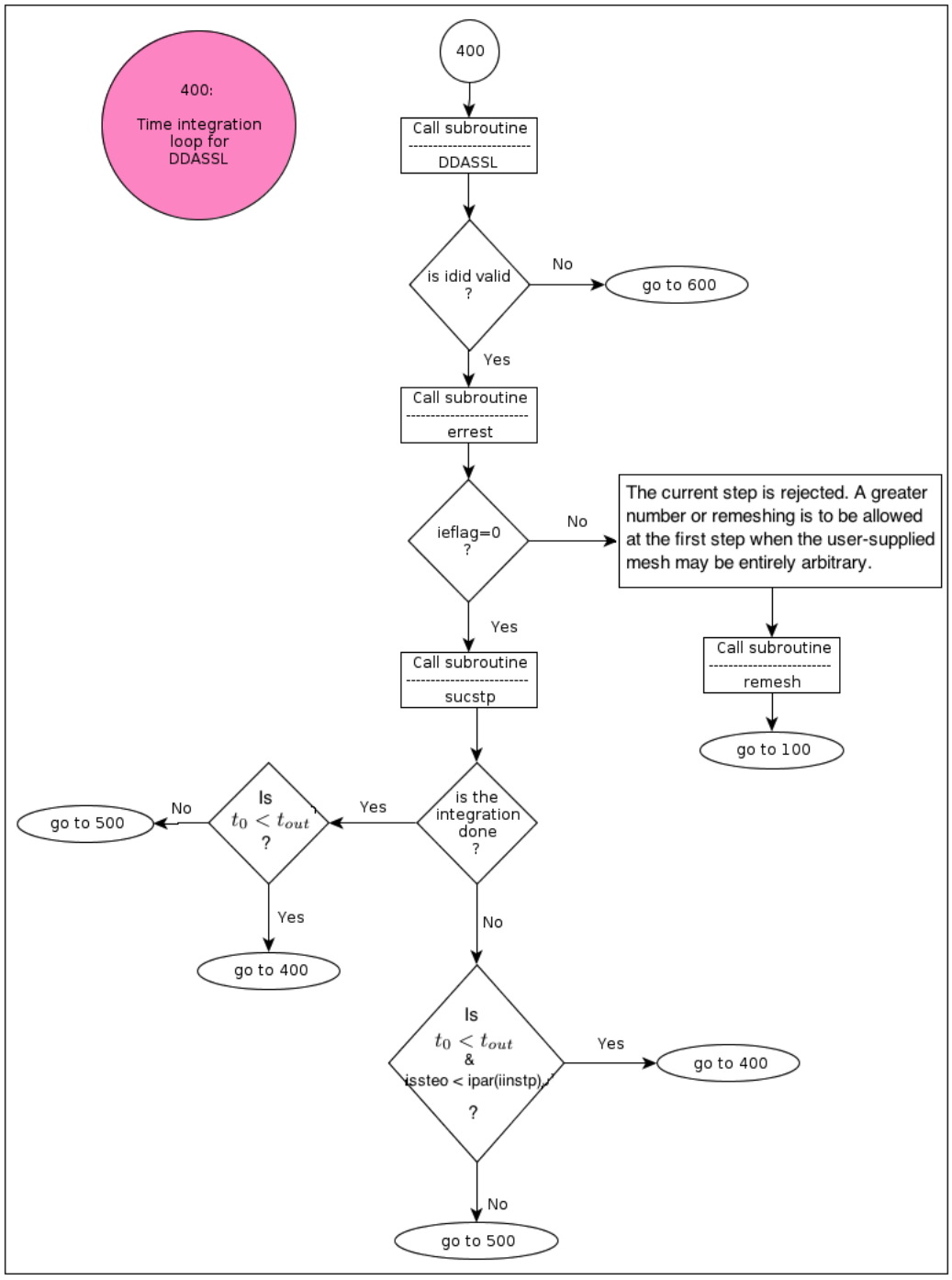

Figure 2.11: Label 400 of the `BACOLI` algorithm.

2.6 Modifications to the BACOLI software package

This section presents the modifications to the BACOLI software package. All of the modifications are done to extend the BACOLI software package to solve the equations that do not have any dependency on spatial derivatives. The system of the equations is

$$\begin{aligned} \boldsymbol{u}_t(x,t) &= \boldsymbol{f}(t,x,\boldsymbol{v}(x,t),\boldsymbol{u}(x,t),\boldsymbol{u}_x(x,t),\boldsymbol{u}_{xx}(x,t)), \\ \boldsymbol{v}_t(x,t) &= \boldsymbol{g}(t,x,\boldsymbol{v}(x,t),\boldsymbol{u}(x,t)), \\ a \le x \le b, &\quad t \ge t_0, \end{aligned} \tag{2.28}$$

the boundary conditions are given by

$$\begin{aligned} \boldsymbol{\mathfrak{B}}_a(t,\boldsymbol{v}(a,t),\boldsymbol{u}(a,t),\boldsymbol{u}_x(a,t)) &= \boldsymbol{0}, \\ \boldsymbol{\mathfrak{B}}_b(t,\boldsymbol{v}(b,t),\boldsymbol{u}(b,t),\boldsymbol{u}_x(b,t)) &= \boldsymbol{0}, \end{aligned} \tag{2.29}$$

and the initial conditions are given by

$$\begin{aligned} \boldsymbol{u}(x,t_0) &= \boldsymbol{u}_0(x), \\ \boldsymbol{v}(x,t_0) &= \boldsymbol{v}_0(x), \end{aligned} \tag{2.30}$$

where $\boldsymbol{u}(x,t)$ and $\boldsymbol{v}(x,t)$ are the unknown vector functions. In this system of equations, the unknown vector function $\boldsymbol{v}(x,t)$ corresponds to the PDEs whose RHS have no spatial derivative. In its present form, the extended BACOLI software package can only solve uncoupled systems of PDEs. The details of the subroutines of the BACOLI software package that were modified are presented as follows.

A list of six modified subroutines is provided together with the tasks that are done by each subroutine. A description of the modifications in each subroutine is also provided. To make the descriptions tangible, a particular case of the system of equations is considered.

The system considered for our descriptions of the modifications is as follows: Consider the system of equations

$$u_t(x,t) = f(t,x,v(x,t),u(x,t),u_x(x,t),u_{xx}(x,t)), \tag{2.31a}$$

$$v_t(x,t) = g(t,x,v(x,t),u(x,t)), \tag{2.31b}$$

that only consists of two equations. The second equation does not have any dependence on spatial derivatives.

In this case, the structure of the ABD matrix is shown in Figure 2.12. In this figure, the four interior blocks indicate that there are four subintervals. In other words, N_x in (2.15) is 5. Each subinterval consists of five collocation points, meaning that the degree of the B-splines is six. The number of continuity for this B-spline is two and the number of differential equations is two. In this figure, the top and bottom blocks correspond to the left and right boundaries, respectively. Because there are two equations in (2.31), the top and bottom blocks consist of two rows. The first row corresponds to (2.31a) and the second row corresponds to (2.31b).

In the modified `BACOLI` software package, the way in which the elements of the second rows of each block are generated is modified. These elements are generated the same way that the elements in the interior blocks are generated.

The first subroutine in the `BACOLI` software package that is modified is the subroutine `iniy`, which performs the initialization tasks of calculating B-spline basis functions, constructing the ABD matrices, and determining the initial vector of the B-spline coefficients. It determines the initial vector of the B-spline coefficients, $\boldsymbol{y}$, through solving the system $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{d}$, where $\boldsymbol{A}$ is an ABD matrix and $\boldsymbol{d}$ is the vector of the unknown functions at the initial time. The vector $\boldsymbol{d}$ is built up through the initial condition subroutine.

The modifications of this subroutine are related to the case where the boundary conditions are not Dirichlet. In this case, the solution of $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{d}$ is considered as an initial guess, and the iterates are obtained through the Newton iteration (2.12). The top and bottom blocks are built up through the boundary conditions subroutine. In the modifications, the second row of the top and bottom blocks are added to the Newton iteration. The first row of the top and bottom blocks remain unchanged. Those elements of the RHS vector, $\boldsymbol{d}$, that correspond to the second equation at the boundaries also are generated the same way as the elements of $\boldsymbol{d}$ corresponding to the second equation at the interior points. In other words, because there is no boundary condition for the second equation, these elements cannot be generated

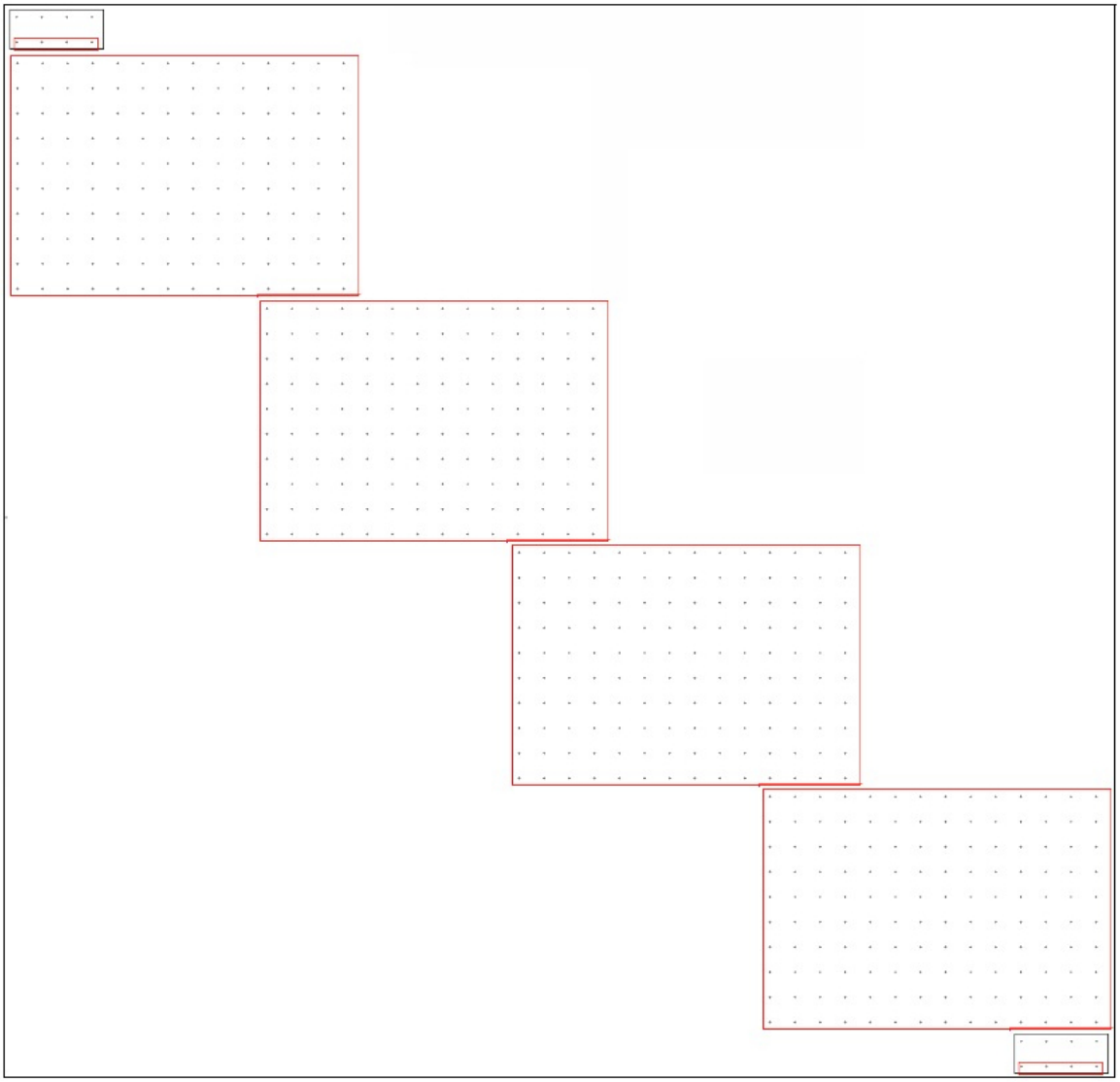

Figure 2.12: ABD matrix structure for the case where the system consists of two equations, the second one of which does not have spatial derivative. In this case, the degree of the B-splines is six.

through boundary condition subroutines.

The second subroutine in the `BACOLI` software package that is modified is the subroutine `iniyp`. This subroutine determines $\boldsymbol{y'}(t_0)$, i.e., the initial vector of B-spline coefficients for the first temporal derivative of $\boldsymbol{y}(t)$. Because the system that is solved in this subroutine is similar to the one solved in the subroutine `iniy`, the modifications of this subroutine are similar to those of the subroutine `iniy`.

The third subroutine in the `BACOLI` software package that is modified is the subroutine `caljac`. The Jacobian of the system of DAEs is determined through this subroutine. Consider a discretized DAE that is linear and of the form $\boldsymbol{Ay'} - \boldsymbol{f} = \boldsymbol{0}$, where the matrix $\boldsymbol{A}$ contains the collocation equations and some boundary conditions information. The vector $\boldsymbol{f}$ contains the RHS of the collocation equations and the corresponding boundary condition. The subroutine `caljac` returns the matrix $\boldsymbol{B} = \frac{d\boldsymbol{G}}{d\boldsymbol{y}} + c_j \frac{d\boldsymbol{G}}{d\boldsymbol{y'}}$, where the (i, j) element of the matrix $\boldsymbol{B}$ involves the partial derivative of equation i with respect to the variable j. The scalar c_j is chosen by the solver `DASSL` to accelerate the convergence of the Newton iteration used to solve the implicit equations resulting from the BDF methods. According to the discretized form of the DAE, $\boldsymbol{B}$ can be written

$$\boldsymbol{B} = c_j \boldsymbol{A} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}, \tag{2.32}$$

where $\boldsymbol{A}$ and $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}$ are ABD matrices. These two matrices are modified in order to be adapted to (2.28). In the standard `BACOLI` software package, the top and bottom blocks of the matrix $\boldsymbol{A}$ are updated in the subroutine `caljac` while the interior blocks are input of this subroutine. In other words, the interior blocks are not updated within the subroutine `caljac`. In our modifications, the second rows of the top and bottom blocks are not updated within the subroutine `caljac`. In the standard `BACOLI` software package, in the matrix $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}$, the top and bottom blocks are zero. In the modified `BACOLI`, they are changed such that the second rows of top and bottom are updated within the subroutine `caljac` the same way that the elements of the interior blocks are generated. A schematic view of these two matrices after the modifications is represented in Figures 2.13 and 2.14, respectively. As can be seen in these two figures, the second row of the top and bottom blocks are generated

the same way as the interior blocks.

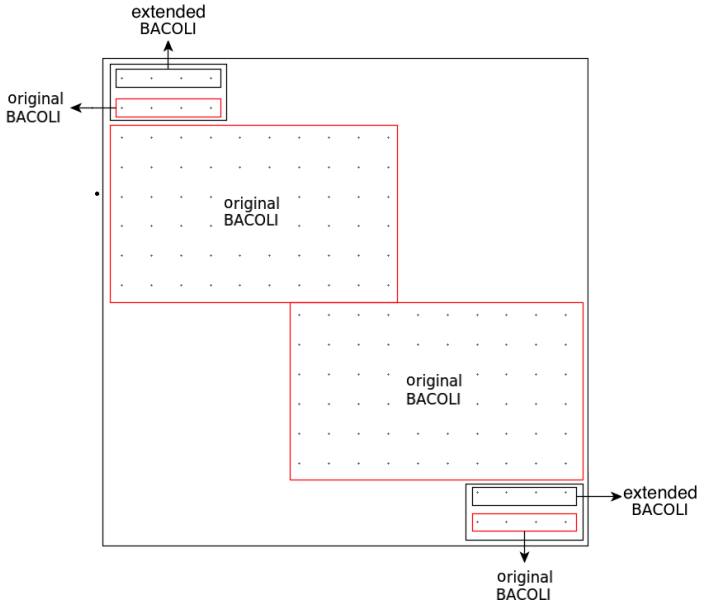

Figure 2.13: ABD matrix $\boldsymbol{A}$ generated in the subroutine `caljac` after the modifications.

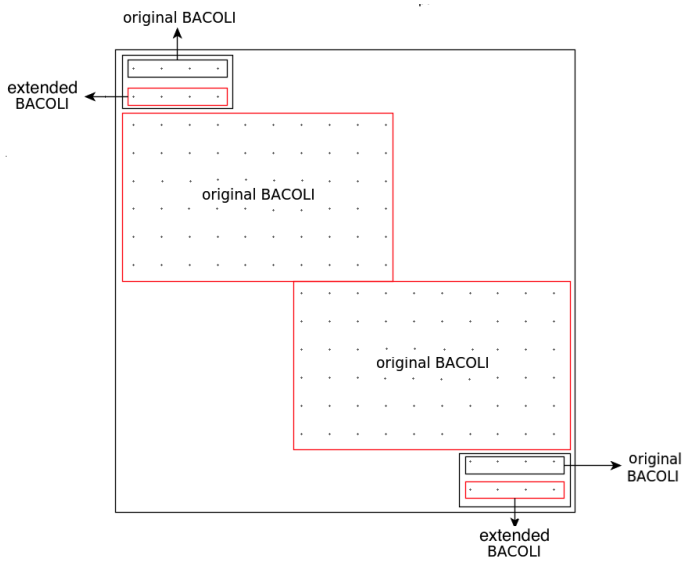

Figure 2.14: ABD matrix $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}$ generated in the subroutine `caljac` after the modifications.

The fourth modified subroutine in the `BACOLI` software package is the subroutine `calres`. This subroutine defines the system of DAEs to be solved by the solver

DASSL. It returns a residual, $\boldsymbol{\delta}$, of the system of DAEs (2.13). The residual of the system, $\boldsymbol{\delta}$, is given by

$$\boldsymbol{\delta} = \boldsymbol{G}(t, \boldsymbol{y}, \boldsymbol{y'}). \tag{2.33}$$

Parts of the vector $\boldsymbol{\delta}$ that correspond to the boundaries are generated through the boundary conditions subroutine. Because of this, the modifications in this subroutine are related to the vector $\boldsymbol{\delta}$. Figure 2.15 shows the modifications of this vector.

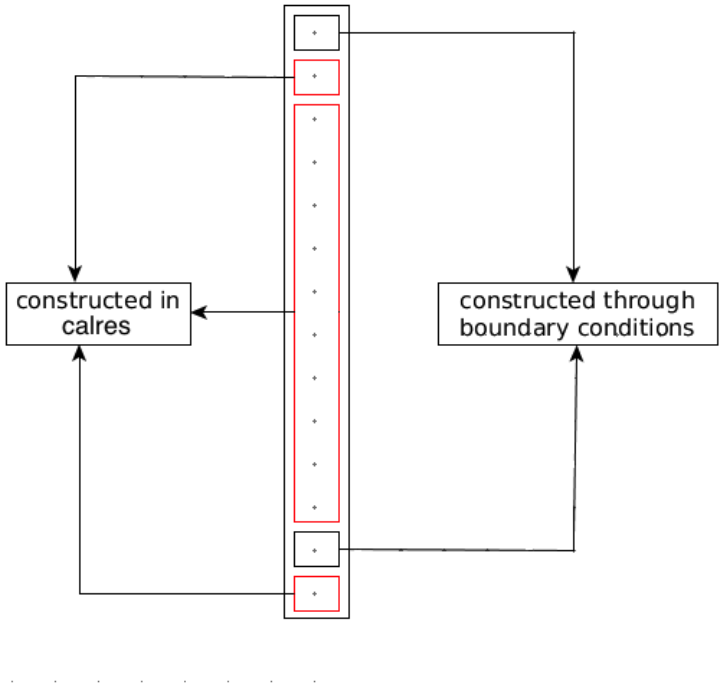

Figure 2.15: The modified vector $\boldsymbol{\delta}$ generated in the subroutine `calres`.

The fifth modified subroutine in the BACOLI software package is the subroutine DDASLV. This subroutine manages the solution of the linear system arising in the Newton iteration. According to Brenan et al. (1996), for a system of DAEs with index α, the condition number of the iteration matrix is $\mathcal{O}(\Delta t^{-\alpha})$, where Δt is the current time step during the integration and α is a positive integer. Thus, when Δt approaches zero, the condition number becomes large. In this case, the corresponding Newton iteration may fail. To overcome this, the conditioning of the Jacobian matrix should be improved. Petzold and Lötstedt Petzold and Lötstedt (1986) proposed a

scaling technique to improve the conditioning of the Jacobian matrix. This technique involves only scaling the RHS elements corresponding to the boundary conditions. In order to adapt this scaling to our case of interest, these parts are limited to the equations with spatial derivatives. Figure 2.16 shows the modifications of the RHS vector $\boldsymbol{\delta}$.

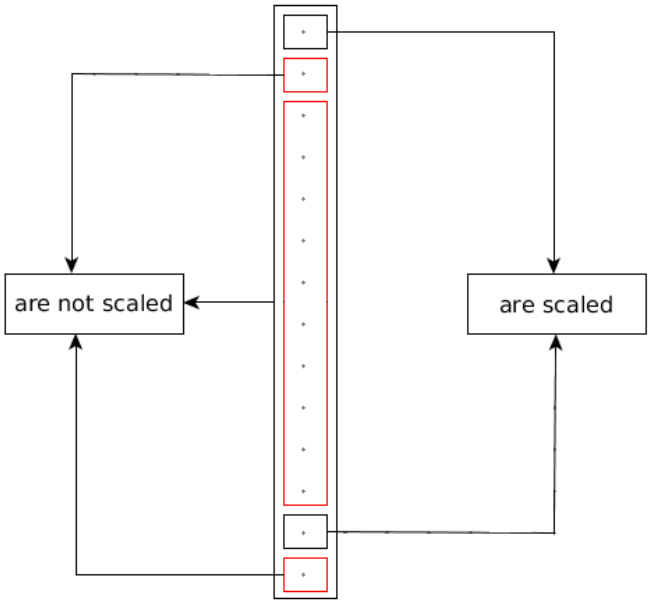

Figure 2.16: The modified vector $\boldsymbol{\delta}$ generated in the subroutine `DDASLV`.

The sixth modified subroutine in the `BACOLI` software package, is the subroutine `errest`. This subroutine computes the error estimates (2.21) and (2.22). It also determines whether a remeshing is required. If a remeshing is required, then the distribution of the new mesh points is determined through this subroutine. No spatial adaptivity is required for (2.31b) nor it is sensible to do so because by definition there is no spatial dependency. The subroutine `errest` is modified based on this.

Chapter 3

Other Related Background

This chapter presents some biological and mathematical background behind the heart models used in this thesis. Followed by a description of some cell models, more details on the monodomain model (1.7) are provided in this chapter.

3.1 Physiology

3.1.1 Heart cycle

The heart is a muscular and chambered organ about the size of a fist, located just behind the breastbone. The heart pumps blood throughout the body. It has four chambers called the right and left atria and the right and left ventricles. The right atrium receives the blood from the veins and pumps it to the right ventricle. The right ventricle receives blood from the right atrium and pumps it to the lungs in order to oxygenate the blood. The left atrium receives oxygenated blood and pumps it to the left ventricle. Finally, the left ventricle pumps the blood throughout the body. This pumping, as a whole, is the consequence of contractions. The contractions are governed by a web of nerves that transmit electrical signals throughout the heart Bailey (2012). Figure 3.1 is a schematic diagram of the heart, adapted from Mackenna and Callander (1997). This diagram represents the four chambers of the heart. The arrows show the direction of the blood as it cycles through different parts of the heart.

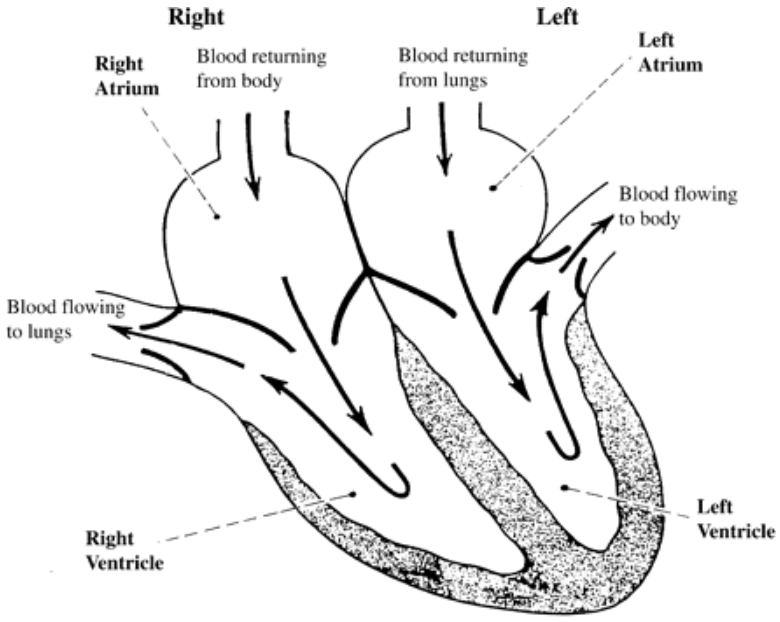

Figure 3.1: The heart as a four-chambered organ Mackenna and Callander (1997).

3.1.2 Electrical analysis of cardiac cells

In this section, some of the properties of the heart from the aspect of the electrical field are discussed. The heart looks like an electric dipole with positive and negative charges that create an electrical field. The body behaves like a conductor when it is exposed to such an electrical field. In each heart cycle, an electrical current runs through the body. This electrical current causes a specific amount of potential at each point of the body, giving rise to a potential difference throughout the body. Considering some specific points on the chest and measuring their potentials with respect to a reference potential, their potential variations during a heart cycle represent the electrical activation of the heart muscle cells. This method of representation is called the electrocardiogram (ECG) Sundnes et al. (2006).

In order to excite a cardiac cell, a stimulus can be exerted. This external stimulus induces an electrical field in the heart. Depending on the strength of the resulting electrical field, there may be some intracellular changes. If the strength of the electrical field is more than a certain level, it will affect the conductive properties of the cell. This effect results in *depolarization*, meaning that the neighbouring positive

ions move into the cell membrane. After depolarization, there is another phase called *repolarization*. In this phase, the cell regains its initial electrical potential. In some special cases, there is a certain time span between depolarization and repolarization; this time span is called the *plateau phase*.

In a nutshell, after the cell excitation there is a three-phase cycle. This cycle includes first the depolarization phase, second the plateau phase, and third the repolarization phase. This cycle is called an *action potential*. Figure 3.2 represents a plot of this cycle for the specific cell model of Bondarenko Bondarenko et al. (2004); Institute (2013). This figure shows recorded variations of the transmembrane potential at different time steps. In this figure, the vertical axis represents the transmembrane potential (in mV) and the horizontal axis shows the corresponding time (in ms).

Signal propagation and its effects on the polarizations of the cells in the atrium and the ventricle are as follows. First, the signal is initiated from a node called the *sinoatrial node* located on the wall of the right atrium. When the signal reaches the atrium, it causes contraction in the atrium, and as a consequence, the ventricle takes the blood in. On the other hand, when the signal reaches the atrium, because the signal induces an electrical field, the cells in the atrium are depolarized. Once the signal reaches the ventricle, it causes contraction there. At this step, cardiac cells in the ventricle are depolarized whereas in the atrium, the cells are repolarized. This process is repeated and the polarization of atrium and ventricle reverse in each heart cycle. In this way, the electrical charges of the heart swap constantly Sundnes et al. (2006).

3.2 Simulation

This section is on the simulation and the three main steps that are required for simulating a system. Given a system such as biological system or more specifically the mechanism of the heart, the first step of the simulation is modelling. In order to solve a problem with some mathematical approaches, we first require to have

that problem in mathematical language. In other words, the system of differential equations that govern the mechanism is derived. An instance of the heart model is (1.7).

Once the mathematical model is derived, the second step of the simulation, namely the discretization, is implemented on the model. Because a complex heart model cannot be solved analytically, one resorts to the numerical approaches. The numerical approaches include discretization of the equations in time and space. The result of the discretization is that the numerical values of the unknown functions at some specific meshes are found. The last step of simulation is the visualization. In a nutshell, the three steps include the numerical modelling, the discretization, and the visualization.

Figure 3.3 represents a schematic view of these three steps. The details of each of them are illustrated in the remainder of this section.

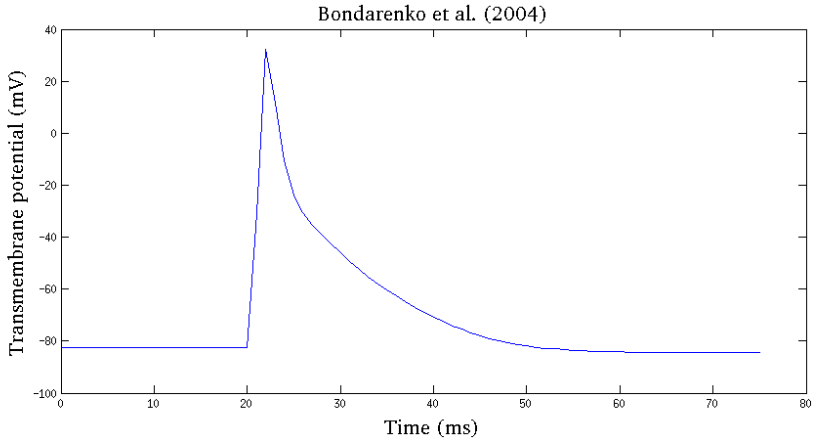

Figure 3.2: Action potential in the model of Bondarenko Bondarenko et al. (2004); Institute (2013).

In order to investigate how a process can occur as a result of activities and interactions on a smaller scale, having detailed information is crucial. In the case of the heart simulation, small scale means the scale at which intracellular activities are considered, as opposed to the large scale where activities in the whole heart system are considered.

Investigating the activity at the small scale and modelling it using some differential equations has two advantages. First, it helps us analyse intracellular electro-

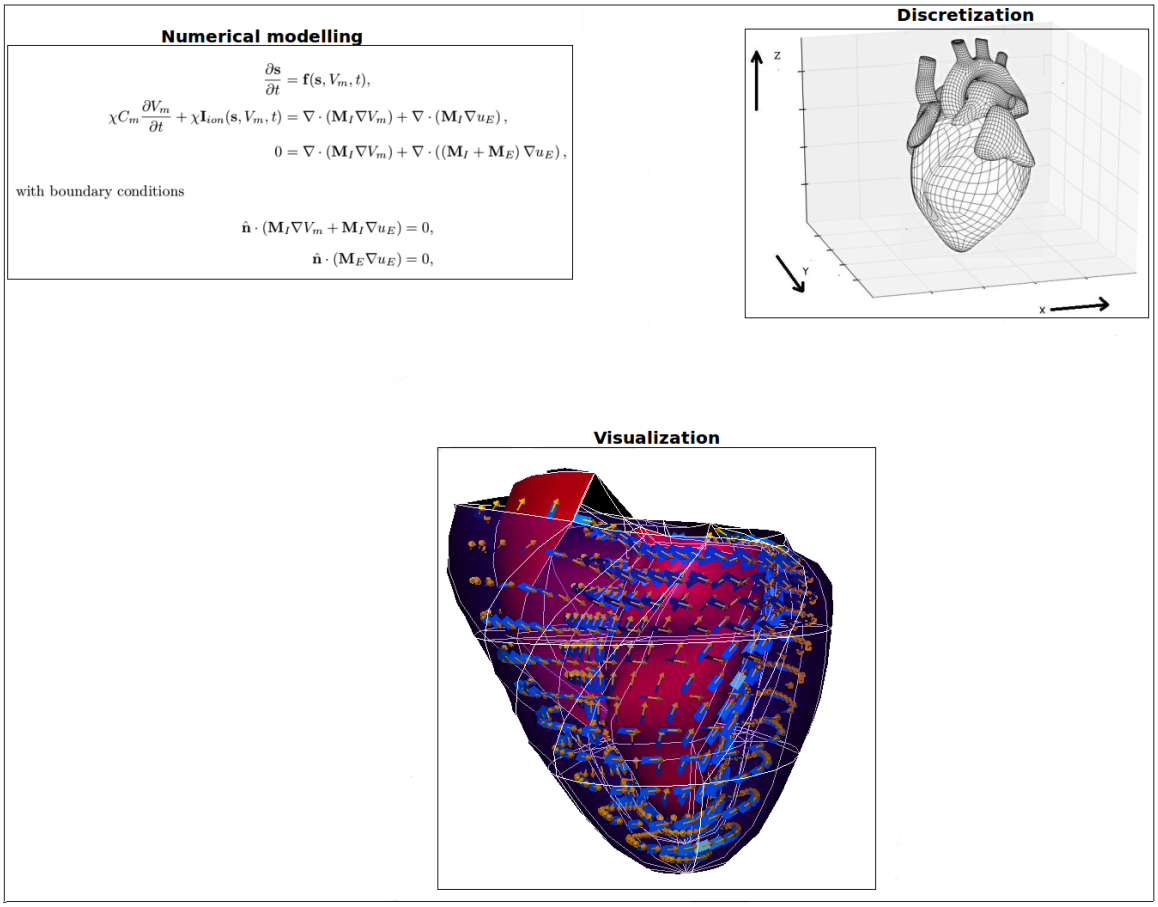

Figure 3.3: Different steps in a numerical simulation.

chemical activities. Second, it casts a light in the path of expressing the whole heart system in a systematic way, as a set of different types of cells. Once we have enough information about smaller parts of the heart, we can formulate its activities.

The model for electrical activity in the heart considered in this thesis is a multi-scale model known as the monodomain model Sundnes et al. (2006). It simulates the evolution of the electrical potential in cardiac tissue and couples the ionic currents at the cellular scale with their propagation at the tissue scale. There are many mathematical models for cardiac cells, e.g., the FitzHugh–Nagumo FitzHugh (1961); Nagumo et al. (1962), Luo–Rudy I Luo and Rudy (1991), and the epicardial variant of ten Tusscher et al. (2006) ten Tusscher and Panfilov (2006) models. Ideally, the more variables that describe different states of the heart, the more realistic the model that is achieved.

Once the mathematical model is obtained, one can solve the differential equations on a computer by means of an appropriate numerical approach. To this end, discretizations of the resulting equations are required in both the time and space dimensions. After the discretizations, the numerical values of the unknown functions at discrete points can be computed.

3.3 Modelling

A *control volume* is a fixed region considered in space in order to study the energies or fluids that pass its boundary. Using this concept in the mathematical models of physical processes, one can utilize continuity, momentum, and energy principles. After specifying the control volume and its boundary, one can form the equations that govern the passage of any fluid or energy through the boundary LLC (2014).

Continuum modelling of a tissue is a method of modelling based on the concept of the control volume. In this type of modelling, some specific sample points are considered. Each of these points is related to its neighbouring volume that consists of some other cells. Each point has a quantity that is the average of its corresponding volume of cells. In continuum modelling, instead of considering individual cells, only

those sample points are considered.

3.3.1 Cardiac cell models

There are many cardiac cell models, from simple to biophysically detailed. There are two advantages of the simple models. First, the actual behaviour of each model parameter can be more easily observed. Second, analysing simple models can give insight into studying biophysically detailed models. However, when it comes to practical point of view there is a considerable disadvantage of simple models. Simple models usually do not provide accurate data. On the other hand, biophysically detailed models that involve more parameters and more details typically provide more accurate and realistic data.

Models of the cardiac cells are capable of representing underlying cardiac electrical dynamics to provide insight into the simulation of the action potential. Depending on the complexity of the model, it may incorporate the formulation of intracellular current voltage, ionic concentration, and ionic channel kinetics responsible for the current. Simple families of cardiac cell models come from the FitzHugh–Nagumo (FHN) model. Generally, one can consider this model for excitable media, and it can be applied to different systems. It consists of two ODEs that are then represented as non-spatially dependent PDEs in the monodomain model.

The FitzHugh–Nagumo model

The FitzHugh–Nagumo model focuses on excitability properties of the cardiac cells and is a modified version of the Van Der Pol equations. The basic idea behind this model is to represent mathematical properties of the cell excitation. These mathematical properties are derived from the electrochemical properties of potassium and sodium ion flow. The model is as follows Sundnes et al. (2006):

$$\begin{aligned} \frac{dv}{dt} &= c_1 v(v-a)(1-v) - c_2 W + I_{stim}, \\ \frac{dW}{dt} &= b(v - c_3 W), \end{aligned} \tag{3.1}$$

where v is the membrane potential, W is the recovery variable, and I_{stim} is the magnitude of a stimulus current. The values of the five parameters a, b, c_1, c_2, and c_3 are presented in Table 3.1. The given values are for the original formulation of the FitzHugh–Nagumo model.

Table 3.1: The values of the five parameters used in the original FitzHugh–Nagumo cell model.

Parameter	Value	Unit
a	0.13	dimensionless
b	0.013	dimensionless
c_1	0.26	ms^{-1}
c_2	0.1	ms^{-1}
c_3	1.0	ms^{-1}

A drawback of this model is that the cell hyperpolarizes in the repolarization phase, whereas in reality there is no such hyperpolarization. A modified version of the FitzHugh–Nagumo model was proposed by Rogers and McCulloch in order to overcome this drawback Sundnes et al. (2006).

The modified version of FitzHugh–Nagumo model is as follows Pitt-Francis et al. (2009):

$$\begin{aligned} \frac{dv}{dt} &= -m_k(v - v_{rest})[W + (v - v_{th})(v - v_{peak})] - I_{stim}, \\ \frac{dW}{dt} &= m_l(v - v_{rest}) - m_b W, \end{aligned} \tag{3.2}$$

where v_{rest}, v_{th}, and v_{peak} are the resting potential, threshold potential, and peak potential respectively. In this study, $v_{th} = -70$ mV, $v_{rest} = -85$ mV, and $v_{peak} = 40$ mV. The constants within our numerical experiments are are presented in Table 3.2.

Table 3.2: The values of the three parameters used in the modified FitzHugh–Nagumo cell model.

Parameter	Value	Unit
m_l	0.63	dimensionless
m_k	4×10^{-4}	ms^{-1}
m_b	0.013	ms^{-1}

The modified version of the FitzHugh–Nagumo model is more realistic than the original one Sundnes et al. (2006).

The Luo–Rudy I model

Another example of cell model is Luo–Rudy I (LR) Luo and Rudy (1991) model. The basic idea in this model is to simulate the action potential of a guinea pig ventricular cell. The eight variables considered in this model include the transmembrane potential, v, the intracellular calcium concentration, Ca_i, and six non-dimensional gating variables, m, j, h, f, d, and X. These variables govern the movement of ions across the cell membrane. The differential equations describing the model are as follows:

$$\begin{aligned} \frac{dv}{dt} &= -\frac{1}{C}(I_{Na} + I_{si} + I_K + I_{K1} + I_{Kp} + I_b), \\ \frac{dCa_i}{dt} &= -10^{-4} I_{si} + 0.07(10^{-4} - Ca_i), \\ \frac{dy}{dt} &= \frac{(y_\infty - y)}{\tau_\infty}, \end{aligned} \tag{3.3}$$

where y can be any of the six gating variables m, j, h, f, d, or X, $y_\infty = y_\infty(v)$, and $\tau_\infty = \tau_\infty(v)$. The currents I_{Na}, I_{si}, I_K, I_{K1}, I_{Kp}, and I_b are non-linear functions of v, Ca_i, m, h, j, d, f, and X. More details on this cell model are mentioned in Luo and Rudy (1991). Accurate cell models such as Luo–Rudy I model describe greater detail such as the calcium concentration and the flow of ions in the cell membrane.

The epicardial variant of the model of ten Tusscher et al. (2006)

The epicardial variant of the model of ten Tusscher et al. (2006) ten Tusscher and Panfilov (2006) is a novel model for the human ventricular cells. This model is based on recent experimental measurements of human action potential duration restitution. It is intended to describe details of calcium dynamics to a level that computationally is feasible for modelling. In this cell model, nineteen variables are considered. This model can be used to show that the recovery dynamics of the fast

sodium current is an important factor in the instability. It is also used to show that steep restitution-mediated fibrillation can occur in the human ventricle. More details, in particular the differential equations describing the model, can be found in ten Tusscher and Panfilov (2006).

3.3.2 A model for heart tissue

The monodomain model is a multi-scale mathematical model for the electrical activity in heart tissue. In this model, the dynamics of the transmembrane potential, v, are described. This model is useful in simplified computational research Sundnes et al. (2006). The standard formulation of this model is (1.7). In this study, the constants χ, M_i, and C_m are 1400 cm $^{-1}$, 1.75 mS/cm, and 1 μF/cm^2 respectively.

Chapter 4

Numerical Results

This chapter presents the numerical results of solving different systems of parabolic partial differential equations. A fully coupled system and some heart models are considered for our experiments.

As mentioned in Chapter 2, in the `BACOLI` software package, there are two tolerances, the relative and absolute tolerances. In our experiments, the relative and absolute tolerances are set to be equal. They are 1×10^{-6} for the fully coupled system and 1×10^{-8} for all of the heart models. In Tables 4.1, 4.2, 4.3, 4.5, and 4.6, the matching digits are in bold font.

Recall the discussion given in Chapter 2 about error estimate schemes. Because the SCI scheme is the standard mode, in our numerical experiments of the heart models, the SCI scheme is chosen. To show that the software package also works with the LOI scheme as an alternative option, LOI scheme is chosen for the fully coupled system.

4.1 Reference solutions and measure of error

In order to measure the errors of the numerical solution obtained with the extended `BACOLI` software package, a reference solution is required. In this section, first the process of generating reference solutions in our experiments is put forth. Subsequently, the criterion for measuring the deviations of the numerical solution with respect to the exact solution is given.

A *reference solution* is generated with `Chaste` for all of the heart models. This software applies a semi-implicit method Whiteley (2006) to the monodomain model (1.7)

and the ODEs corresponding to the cell models are solved with Heun's method. Using the semi-implicit method to discretize time in the monodomain model leads to

$$\chi C_m \frac{v^n - v^{n-1}}{\Delta t} + \chi I_{\text{ion}}(\boldsymbol{s}^{n-1}, v^{n-1}, t^{n-1}) = \frac{\lambda}{1+\lambda} \nabla \cdot (\sigma_i \nabla v^n),$$
$$\frac{\boldsymbol{s}^n - \boldsymbol{s}^{n-1}}{\Delta t} = \boldsymbol{f}(\boldsymbol{s}^{n-1}, v^n, t^{n-1}).$$

In this method, at time step n, the spatial derivatives are evaluated at time t_n and the I_{ion} term is evaluated at time t_{n-1}.

The reference solution is generated by comparing increasingly accurate solutions. In order to compute each successive solution, the time step is halved and the number of spatial mesh points is doubled. The process of generating these increasingly accurate solutions continues until four or more matching digits are obtained. In all of our numerical experiments, the comparison is made at 21 equally spaced time steps in the temporal domain and 101 equally spaced spatial points.

The cardiac models considered in this chapter are the monodomain model coupled with the standard FitzHugh (1961); Nagumo et al. (1962) and modified Sundnes et al. (2006) FitzHugh–Nagumo cell models, the Luo–Rudy I cell model Luo and Rudy (1991), and the epicardial variant of the model of ten Tusscher et al. (2006) ten Tusscher and Panfilov (2006). The resolutions required to generate the reference solution for these numerical experiments are as follows. For the standard and modified FitzHugh–Nagumo models, $\Delta t = 5 \times 10^{-7}$ ms and $\Delta x = \frac{1}{20000}$ cm are used. For the Luo–Rudy I model, $\Delta t = 5 \times 10^{-8}$ ms and $\Delta x = \frac{1}{30000}$ cm are used.

In order to measure the accuracy and efficiency of any numerical method, one can compute an average of the error at N_t points in the temporal domain $t \in [t_0, t_{end}]$ and at N_x points in the spatial domain. Therefore, the average is over $N = N_t N_x$ points. After generating the reference solution for all N points, the *mixed root mean square* (MRMS) Marsh et al. (2012) error is computed as follows:

$$e_{\text{MRMS}} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(\frac{\hat{v}_i - v_i}{1 + |\hat{v}_i|} \right)^2},$$

where $\hat{v}_i$ is the reference solution for the voltage and v_i is the numerical solution for the voltage, both at time t^n.

4.2 Coupled system

An example of a system of equations that cannot be solved with the standard BACOLI software package is

$$\begin{aligned} u_t &= xu_{xx} + txu_x + (1+tx)e^{-tx}v, \qquad 1 \le x \le 2, \\ v_t &= \frac{-1}{2}e^{tx}u - \frac{1}{2}\sqrt{1-v^2}, \qquad 0 \le t \le 1, \end{aligned}$$

with the boundary condition given by

$$\begin{aligned} u(1,t) &= e^{-t}\sin(1+t), \\ u_x(2,t) &= -tu(2,t) + e^{-2t}v(2,t), \end{aligned}$$

and the initial condition is

$$\begin{aligned} u(x,0) &= \sin(x), \\ v(x,0) &= \cos(x). \end{aligned}$$

The analytical solution of this fully coupled system is

$$\begin{aligned} u(x,t) &= e^{-tx}\sin(x+t), \\ v(x,t) &= \cos(x+t). \end{aligned}$$

After the modifications, it can be solved with the extended BACOLI software package. With the choice of the LOI error estimate and a tolerance of 1×10^{-6}, approximations to the unknown functions $u(x,t)$ and $v(x,t)$ at $t_{end} = 1$ are given in columns two and four of Table 4.1. In this table, the first column shows the points at which we obtained the solutions. The third and fifth columns show the analytical solution of $u(x,t)$ and $v(x,t)$ at the corresponding points. The matching digits between the numerical and the analytical solutions are shown in bold font.

In this table, because the tolerance is 1×10^{-6}, one should expect about six matching digits between the reference solution and the numerical solution. As can be seen in this table, the number of matching digits at different points is either five or six.

Table 4.1: Solution of the fully coupled system obtained with the extended `BACOLI` software package (LOI scheme and tolerance 1×10^{-6}).

x	Numerical solution of $u(x,1)$	Analytical solution of $u(x,1)$	Numerical solution of $v(x,1)$	Analytical solution of $v(x,1)$
1.0	**0.334511829**6294	**0.334511829**2392	**–0.4161**452687667	**–0.4161**468365471
1.25	**0.222921**8216328	**0.222921**7032167	**–0.62817**23534610	**–0.62817**36227227
1.5	**0.133537**3354119	**0.133537**1853582	**–0.8011**425468139	**–0.8011**436155469
1.75	**0.066322**8873065	**0.066322**7356501	**–0.9243**015129225	**–0.9243**023786324
2.0	**0.019098**6701568	**0.019098**5162611	**–0.98999**19811429	**–0.98999**24966004

4.3 The monodomain model with the FitzHugh–Nagumo cell model

This section provides the numerical results obtained by solving the monodomain model (1.7) coupled with the FitzHugh–Nagumo cell models (3.1) and (3.2) with the extended `BACOLI` software package. Because the cell model is a non-spatially dependent PDE, the original `BACOLI` software package cannot solve the system. After the modifications, the extended `BACOLI` software package was used to solve the monodomain model coupled first with the FitzHugh–Nagumo cell model (3.1) and second with the modified FitzHugh–Nagumo cell model (3.2). The results are as follows.

The system of monodomain model coupled with the original FitzHugh–Nagumo model with the parameters $\psi = 1400$, $\lambda = 1$, and $c_m = 1$ is considered in our experiment. The spatial domain is $[0, 1]$ with the boundary conditions given by

$$\begin{cases} \frac{\partial v}{\partial x}(0,t) = 0, \\ \frac{\partial v}{\partial x}(1,t) = 0, \end{cases}$$

and the initial conditions given by

$$\begin{cases} v(x,0) = 2 + 10(1 - \sin(x)), \\ W(0) = -0.5758. \end{cases}$$

The results of solving the monodomain model coupled with the original FitzHugh–Nagumo model with a continuous initial condition are shown in Table 4.2. The first, second, and third columns contain the spatial points, the reference solution generated with `Chaste`, and the numerical solution for the voltage obtained with the extended `BACOLI` software package, respectively. These results are for the time $t_{end} = 20$ ms.

Table 4.2: Solution of the original FitzHugh–Nagumo for the voltage (mV) after 20 ms, obtained with `Chaste` and Extended `BACOLI` software package (SCI scheme and tolerance 1×10^{-8}).

x	**Reference solution**	**Extended `BACOLI` solution**
0.0	$-$**1.51887**301391762	$-$**1.51887**245743619
0.1	$-$**1.51888**063512694	$-$**1.51888**010507247
0.2	$-$**1.51889**882819236	$-$**1.51889**835832303
0.3	$-$**1.5189**2023977051	$-$**1.5189**1982726814
0.4	$-$**1.51894**537278728	$-$**1.51894**501438495
0.5	$-$**1.51897**489618657	$-$**1.51897**458221901
0.6	$-$**1.51900**953111721	$-$**1.51900**925972157
0.7	$-$**1.5190**5001938151	$-$**1.5190**4978682755
0.8	$-$**1.51909**699478253	$-$**1.51909**679494815
0.9	$-$**1.51915**041733144	$-$**1.51915**024519284
1.0	$-$**1.51917**616997381	$-$**1.51917**600538379

In this table, because the tolerance is 1×10^{-8}, one should expect about eight matching digits between the reference solution and the numerical solution. As can be seen in this table, the number of matching digits at the points $x = 0.3$ and $x = 0.7$ is five. At the point $x = 0.0$, there are six matching digits and at all of the other points there are seven matching digits.

For the system of monodomain model coupled with the modified FitzHugh–Nagumo cell model the results are presented in Table 4.3. In this table, the first column shows the spatial points at which the solutions are shown. The second column contains the reference solution for the voltage obtained with `Chaste`. Because generating reference solution for this model is not computationally demanding, the reference solution has seven matching digits and MRMS error is $3.15 \times 10^{-7}\%$. The reported matching digits and MRMS error during the process of generating reference

solution are between two consecutive phases, where at each phase the time step is halved and the number of spatial mesh points is doubled. The third column contains the numerical solution for the voltage obtained with the extended `BACOLI` software package where the SCI scheme is chosen as the spatial error estimate. These results are for the time $t_{end} = 5$ ms.

Table 4.3: Solution of the modified FitzHugh–Nagumo for the voltage (mV) after 5 ms, obtained with `Chaste` and Extended `BACOLI` software package (SCI scheme and tolerance 1×10^{-8}).

x	**Reference solution**	**Extended `BACOLI` solution**
0.0	−**70.07892**967359772	−**70.07892**970713885
0.1	−**70.11555**741230695	−**70.11555**744141333
0.2	−**70.20802**042405890	−**70.20802**046802686
0.3	−**70.33462**469584382	−**70.33462**470143223
0.4	−**70.49056**952202749	−**70.49056**947955525
0.5	−**70.68216**243633999	−**70.68216**231557231
0.6	−**70.92115**668214646	−**70.92115**654067435
0.7	−**71.22398**332528229	−**71.22398**310861703
0.8	−**71.60154**156739336	−**71.60154**133254113
0.9	−**71.99352**280416010	−**71.99352**258416253
1.0	−**72.18003**449141124	−**72.18003**428847781

In this table, at the point $x = 0.1$ there are nine matching digits between the reference solution and the numerical solution. At all of the other points, there are eight matching digits.

4.4 The monodomain model with the Luo–Rudy I cell model

In this section, the results of solving the monodomain model coupled with the Luo–Rudy I cell model Luo and Rudy (1991) are provided. The system of equations for

the Luo–Rudy I cell model is as follows:

$$
\begin{aligned}
\frac{dv}{dt} &= -\frac{1}{C_m}(I_{\text{ion}} + I_{st}),\\
\frac{dm}{dt} &= \alpha_m(1-m) - \beta_m m,\\
\frac{dh}{dt} &= \alpha_h(1-h) - \beta_h h,\\
\frac{dj}{dt} &= \alpha_j(1-j) - \beta_j j,\\
\frac{dd}{dt} &= \alpha_d(1-d) - \beta_d d,\\
\frac{df}{dt} &= \alpha_f(1-f) - \beta_f f,\\
\frac{d\left([\text{Ca}]_i\right)}{dt} &= -10^{-4} I_{\text{si}} + 0.07(10^{-4} - [\text{Ca}]_i),\\
\frac{dX}{dt} &= \alpha_X(1-X) - \beta_X X,
\end{aligned}
\tag{4.1}
$$

where v is the transmembrane potential for an individual cardiac cell Luo and Rudy (1991), m, h, j, d, f, $[\text{Ca}]_i$, and X are activation gate, fast inactivation gate, slow inactivation gate, activation gate, inactivation gate, calcium uptake, and activation gate, respectively. In equations (4.1), the parameter C_m is the membrane capacitance and I_{st} is the stimulus current applied by the sinoatrial node. The I_{ion} term is the total ionic current that is defined as follows:

$$
\begin{aligned}
I_{\text{ion}} &= I_{\text{Na}} + I_{\text{si}} + I_{\text{K}} + I_{\text{K1}} + I_{\text{Kp}} + I_{\text{b}}\\
&= \overline{G}_{\text{Na}} \cdot m^3 \cdot h \cdot j \cdot (v - E_{\text{Na}}) + \overline{G}_{\text{si}} \cdot d \cdot f \cdot (v - E_{\text{si}})\\
&\quad + \overline{G}_{\text{K}} \cdot X \cdot X_{\text{i}} \cdot (v - E_{\text{K}}) + \overline{G}_{\text{K1}} \cdot \text{K1}_{\text{inf}} \cdot (v - E_{\text{K1}})\\
&\quad + \overline{G}_{\text{KP}} \cdot \text{Kp} \cdot (v - E_{\text{Kp}}) + \overline{G}_{\text{b}} \cdot (v - E_{\text{b}}).
\end{aligned}
\tag{4.2}
$$

All of the parameters that appear in equations (4.1) and (4.2) are defined as follows:

$$I_{\mathrm{Na}} = \overline{G}_{\mathrm{Na}} \cdot m^3 \cdot h \cdot j \cdot (v - E_{\mathrm{Na}}),$$

$$\alpha_m = \frac{0.32(v + 47.13)}{1 - e^{-0.1(v+47.13)}},$$

$$\beta_m = 0.08e^{-v/11},$$

$$\alpha_h = \begin{cases} 0.135e^{(v+80)/-6.8}, & v < -40 \text{ mV}, \\ 0, & v \geq -40 \text{ mV}, \end{cases}$$

$$\beta_h = \begin{cases} 3.56e^{0.079v} + 3.1 \cdot 10^5 e^{0.35v}, & v < -40 \text{ mV}, \\ \dfrac{1}{0.13(1 + e^{(v+10.66)/-11.1})}, & v \geq -40 \text{ mV}, \end{cases}$$

$$\alpha_j = \begin{cases} \dfrac{-1.2714 \cdot 10^5 e^{0.2444v} - 3.474 \cdot 10^{-5} e^{-0.04391v} \cdot (v + 37.78)}{1 + e^{0.311(v+79.23)}}, & v < -40 \text{ mV}, \\ 0, & v \geq -40 \text{ mV}, \end{cases}$$

$$\beta_j = \begin{cases} \dfrac{0.1212e^{-0.01052v}}{1 + e^{-0.1378(v+40.14)}}, & v < -40 \text{ mV}, \\ \dfrac{0.3e^{-2.535 \cdot 10^{-7} v}}{1 + e^{-0.1(v+32)}}, & v \geq -40 \text{ mV}, \end{cases}$$

$$I_{\mathrm{si}} = \overline{G}_{\mathrm{si}} \cdot d \cdot f \cdot (v - E_{\mathrm{si}}),$$

$$E_{\mathrm{si}} = 7.7 - 13.0287 \cdot \ln\left([\mathrm{Ca}]_i\right),$$

$$\alpha_d = \frac{0.095e^{-0.01(v-5)}}{1 + e^{-0.072(v-5)}},$$

$$\beta_d = \frac{0.07e^{-0.017(v+44)}}{1 + e^{0.05(v+44)}},$$

$$\alpha_f = \frac{0.012e^{-0.008(v+28)}}{1 + e^{0.15(v+28)}},$$

$$\beta_f = \frac{0.0065e^{-0.02(v+30)}}{1 + e^{-0.2(v+30)}},$$

$$I_{\mathrm{K}} = \overline{G}_{\mathrm{K}} \cdot X \cdot X_{\mathrm{i}} \cdot (v - E_{\mathrm{K}}),$$

$$\overline{G}_{\mathrm{K}} = 0.282 \cdot \sqrt{[\mathrm{K}]_o / 5.4},$$

$$
\begin{aligned}
\alpha_X &= \frac{0.0005e^{0.083(v+50)}}{1+e^{0.057(v+50)}},\\
\beta_X &= \frac{0.0013e^{-0.06(v+20)}}{1+e^{-0.04(v+20)}},\\
X_{\mathrm{i}} &= \begin{cases} \dfrac{2.837(e^{0.04(v+77)}-1)}{(v+77)e^{0.04(v+35)}}, & v > -100 \text{ mV},\\ \\ 1, & v \leq -100 \text{ mV}, \end{cases}\\
I_{\mathrm{K1}} &= \overline{G}_{\mathrm{K1}} \cdot \mathrm{K1}_\infty \cdot (v - E_{\mathrm{K1}}),\\
\overline{G}_{\mathrm{K1}} &= 0.6047 \cdot \sqrt{[\mathrm{K}]_o/5.4},\\
\mathrm{K1}_\infty &= \frac{\alpha_{\mathrm{K1}}}{\alpha_{\mathrm{K1}}+\beta_{\mathrm{K1}}},\\
\alpha_{\mathrm{K1}} &= \frac{1.02}{1+e^{0.2385(v-E_{\mathrm{K1}}-59.215)}},\\
\beta_{\mathrm{K1}} &= \frac{0.49124e^{0.08032(v-E_{\mathrm{K1}}+5.476)}+e^{0.06175(v-E_{\mathrm{K1}}-594.31)}}{1+e^{-0.5143(v-E_{\mathrm{K1}}+4.753)}},\\
I_{\mathrm{Kp}} &= \overline{G}_{\mathrm{KP}} \cdot \mathrm{Kp} \cdot (v - E_{\mathrm{Kp}}),\\
E_{\mathrm{Kp}} &= E_{\mathrm{K1}},\\
\mathrm{Kp} &= \frac{1}{1+e^{(7.488-v)/5.98}},\\
I_{\mathrm{b}} &= \overline{G}_{\mathrm{b}} \cdot (v - E_{\mathrm{b}}).
\end{aligned}
$$

Table 4.4 shows the values of the channel conductances, the reversal potentials for the ions and other parameters.

Table 4.5 shows the results of solving the system of monodomain model coupled with the Luo–Rudy I cell model. The first column contains the spatial points at which the output is computed. The second column in this table, represents the reference solution generated by `Chaste`. This reference solution has four matching digits and MRMS error $4.86 \times 10^{-4}\%$ between two consecutive phases, where at each phase the time step is halved and the number of spatial mesh points is doubled. The third column represents a numerical solution generated with the extended `BACOLI` software package.

As can be seen in this table, the number of matching digits at the points 0.0 and 0.1 is seven, and at all of the other points it is six.

Table 4.4: Different parameters for the Luo–Rudy I cell model Clements (1996).

Channel conductance (ms/cm^2)		
$\overline{G}_{\mathrm{Na}}$	=	23.0
$\overline{G}_{\mathrm{si}}$	=	0.09
$\overline{G}_{\mathrm{K}}$	=	0.282
$\overline{G}_{\mathrm{K1}}$	=	0.6047
$\overline{G}_{\mathrm{Kp}}$	=	0.0183
$\overline{G}_{\mathrm{b}}$	=	0.03921
Reversal potential (mV)		
E_{Na}	=	54.4
E_{si}	=	118.7
E_K	=	–77
E_{K1}	=	–87.2
E_{Kp}	=	–87.2
E_b	=	–59.87
Other parameters		
Resting Membrane Potential v_{rest} = –84.0 mV		
Membrane Threshold Potential $v_{threshold}$ = –60 mV		
$[K]_o$ = 5.4 mM		
Membrane Capacitance C_m = 1 μF/cm^2		

Table 4.5: Solution of the Luo–Rudy I for the voltage (mV) after 5 ms, obtained with `Chaste` and Extended `BACOLI` software package (SCI scheme and tolerance 1×10^{-8}).

x	**Reference solution**	**Extended `BACOLI` solution**
0.0	**19.72929**206746164	**19.72929**17671303982
0.1	**19.41839**006702326	**19.41839**58525026057
0.2	**18.86929**313940373	**18.8692**734805216737
0.3	**18.64696**199620707	**18.6469**465874087490
0.4	**18.85464**593756643	**18.8546**280990272841
0.5	**19.23274**161522291	**19.232**6861315072790
0.6	**19.29529**722142660	**19.2952**427151648926
0.7	**18.32806**198648634	**18.32**79928783199466
0.8	**16.61051**308307339	**16.610**4957169274741
0.9	**17.54827**114618889	**17.5482**387840178511
1.0	**20.52934**860573765	**20.5293**347833165818

In order to show an average behaviour of the three solutions in all of the time steps, the averages of the solutions at the 21 equally spaced spatial points are computed. For example, on the curve obtained by the extended `BACOLI` software package, consider a point $(x_i, \bar{v}_i)$. The second coordinate of this point, $\bar{v}_i$, represents an average of the solutions at x_i in all of the 21 time steps. The average was derived for the reference solution, the numerical solution by the extended `BACOLI` software package with MRMS error 4.15%, and the other numerical solution generated by `Chaste` with MRMS error 4.76%. In order to obtain the numerical solution by the extended `BACOLI` software package with MRMS error 4.15%, the number of collocation points is 3, and the absolute and relative tolerances are 1.25×10^{-2}. This tolerance is roughly the largest tolerance that gives a numerical solution with MRMS error of about 5%. Because it is possible to achieve MRMS error of about 5% with relatively large tolerances, it is most efficient for the extended `BACOLI` software package to use a relatively small number of collocation points. The initial mesh is set to have 4 uniform subintervals in order to minimize any bias in the extended `BACOLI` software package determining a suitable mesh. Figure 4.1 represents a conceptual idea of what 5% MRMS error means. In fact, because the three curves in Figure 4.1 are

close to each other, the solution with 5% MRMS error is quite accurate.

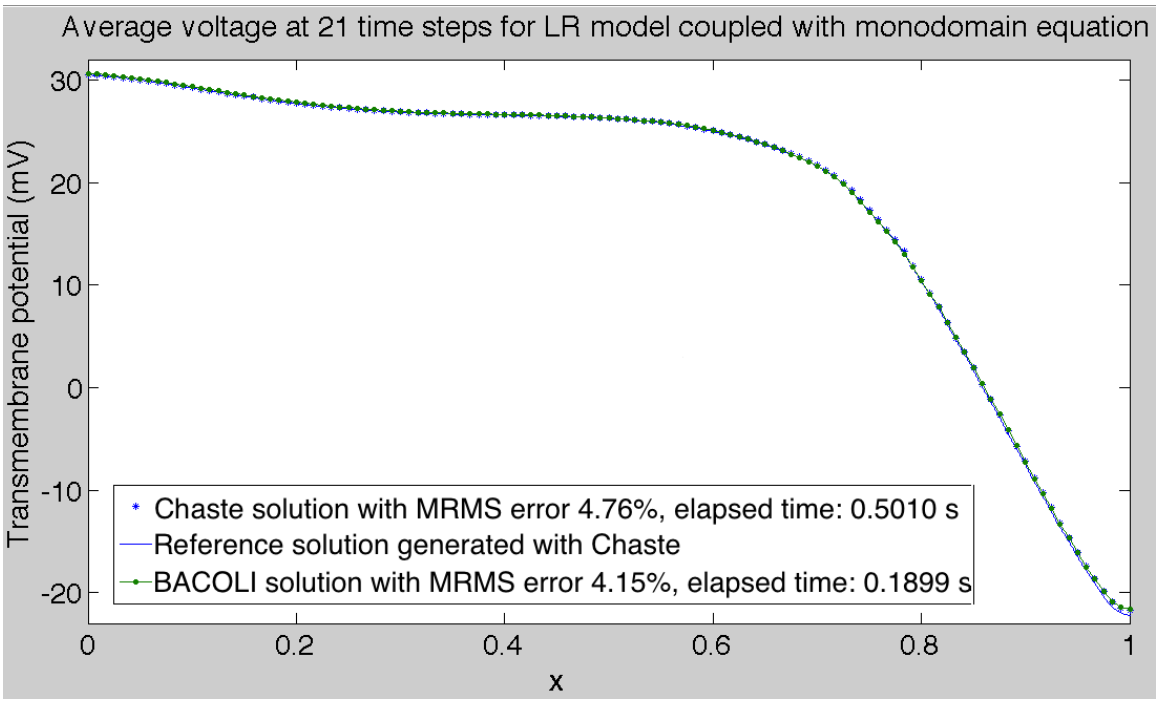

Figure 4.1: Solution of the monodomain model coupled with the Luo–Rudy I model for the voltage with around 5% MRMS error and the elapsed time.

In order to distinguish between the three solutions, Figure 4.2 shows a zoomed-in region of the Figure 4.1.

A comparison of elapsed time between the extended `BACOLI` software package and `Chaste` is made. In these comparisons, the reported elapsed time is the minimum of elapsed times of ten runs. As can be seen in Figure 4.1, the time required to obtain the numerical solution by the extended `BACOLI` software package is 0.18 seconds. The time required to obtain the numerical solution by `Chaste` with almost the same MRMS error is 0.50 seconds. Therefore, in this case there is a speed-up of a factor of around three in the extended `BACOLI` software package. Although the actual amount of time saved on one run may not seem significant, this saving becomes significant when such simulations are run many times; for example, running the Luo–Rudy I model a million times consecutively with the extended `BACOLI` software package saves about $320,000$ seconds or 3.7 days.

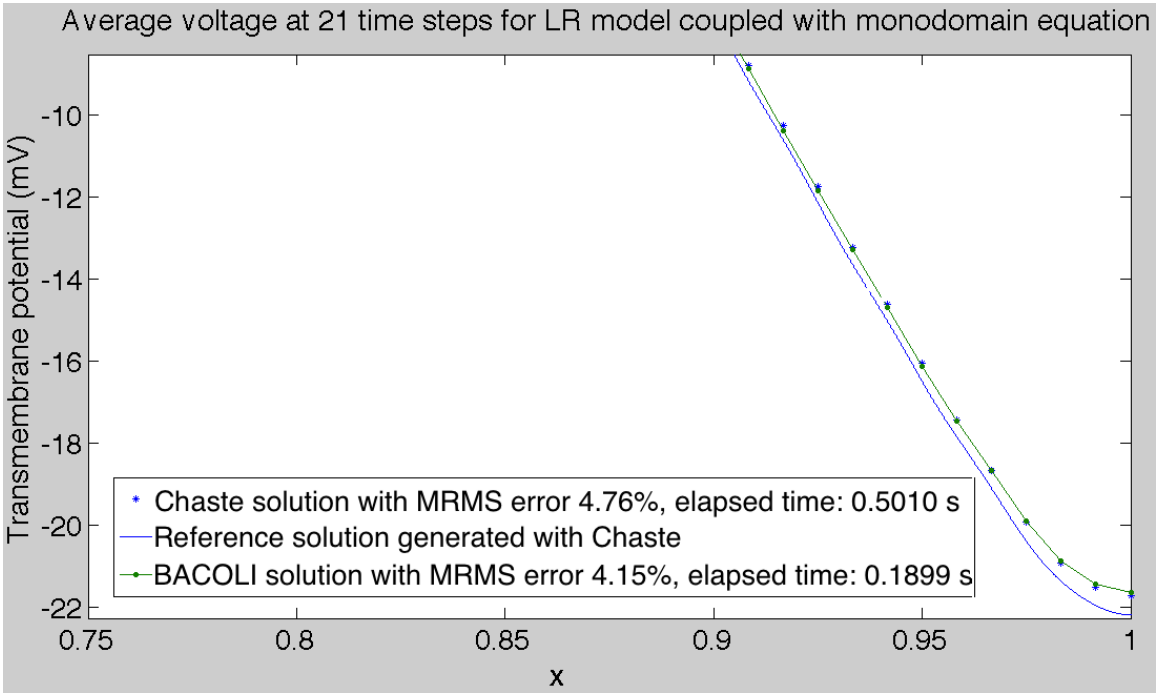

Figure 4.2: Zoomed-in solution of the monodomain model coupled with the Luo–Rudy I model for the voltage with around 5% MRMS error.

4.5 The monodomain model with the epicardial variant of the model of ten Tusscher et al. (2006)

In this section, the results of solving the monodomain model coupled with the epicardial variant of the model of ten Tusscher et al. (2006) are provided. This model is used to simulate the human ventricle. It consists of 19 variables. More details on this model are provided in ten Tusscher and Panfilov (2006). Table 4.6 shows these results. The first column of this table contains the spatial points at which the output is computed. The second column of this table represents the reference solution generated by `Chaste`. This reference solution has three matching digits and MRMS error $3.92 \times 10^{-3}\%$ between two consecutive phases where at each phase the time step is halved and the number of spatial mesh points is doubled. The third column of this table represents the solution generated with the extended `BACOLI` software

package. To obtain this solution, the tolerance is set to be 1×10^{-8} and the spatial error estimate is SCI. The results are reported at the time $t_{end} = 5$ ms.

Table 4.6: Solution of the epicardial variant of the model of ten Tusscher et al. (2006) for the voltage (mV) after 5 ms, obtained with `Chaste` and Extended `BACOLI` software package (SCI scheme and tolerance 1×10^{-8}).

x	**Reference solution**	**Extended `BACOLI` solution**
0.0	**15.736**71511358962	**15.736**6969196549213
0.1	**15.7695**8211267218	**15.7695**645163427329
0.2	**15.8815**3751011188	**15.8815**217931186137
0.3	**15.9865**252395785	**15.9865**116935127158
0.4	**16.078**90988375955	**16.078**8979751286440
0.5	**16.2070**3958867538	**16.2070**297987575209
0.6	**16.401**15130209244	**16.401**0866279204457
0.7	**16.690**85199093823	**16.690**2710748338166
0.8	**17.36**927102074231	**17.36**68857822675768
0.9	**19.56233**91278139	**19.56233**60899577428
1.0	**20.7**3937330189026	**20.7**486443492494281

A comparison of elapsed time between the extended `BACOLI` software package and `Chaste` is made. This comparison is made between two numerical solutions. For the numerical solution obtained with the extended `BACOLI` software package, the absolute and relative tolerances are 1×10^{-2}. Again because these tolerances are relatively large, the number of collocation points is chosen to be three. This tolerance is roughly the largest tolerance that gives a numerical solution with MRMS error of about 5%. For consistency with the previous experiment, the initial mesh size is taken equal to 4. The resulting numerical solution has MRMS error 4.53%. For the numerical solution obtained with `Chaste`, the MRMS error is 5.60%. The results show that with the extended `BACOLI` software package it takes 1.16 seconds and with `Chaste` it takes 2.58 seconds. Therefore, in this case there is a speed-up of a factor of around two in the extended `BACOLI` software package.

Chapter 5

Conclusion

5.1 Summary

In this thesis, the BACOLI software package is studied. Adaptivity in space and time as well as applying a high-order method for solving the systems are the features of the BACOLI software package. The focus in this thesis was to make some modifications to this high-order numerical software package. The purpose of our modifications was to change the BACOLI software package such that it solves a broader spectrum of equations that includes the equations that have no dependency on spatial derivatives.

The practical application of multi-scale problems used in this thesis is mathematical models of the electrical activity of the heart. These models consist of two scales, the cellular and tissue scales.

The monodomain model, coupled with a cardiac cell model that describes the cellular electrical activity of a cardiac cell, describes the propagation of the electrical activity throughout the heart. The models that we considered in this thesis include the one-dimensional monodomain model coupled with the original FitzHugh–Nagumo cell model, modified FitzHugh–Nagumo cell model, Luo–Rudy I cell model, and the epicardial variant of the model of ten Tusscher et al. (2006).

In Chapter 2, the structure of the BACOLI software package was presented. A schematic view of its general algorithm was presented in Figures 2.7, 2.8, 2.9, 2.10, and 2.11. All of the modifications we made to extend the BACOLI software package were mentioned in detail in this chapter.

In Chapter 3, some electrophysiological properties of the heart were studied. An explanation of the FitzHugh–Nagumo, Luo–Rudy I, and epicardial variant of the

model of ten Tusscher et al. (2006) models together with the monodomain model were put forth in this chapter.

The results of our research were described in Chapter 4. In this chapter, two types of results were described. First, we showed that after the modifications of the BACOLI software package, it can solve a new class of equations. This category of equations is represented by (2.28), (2.29), and (2.30). Second, the speed-up of the extended BACOLI software package is considered. We compared the time elapsed to obtain the solution of the monodomain model coupled with the Luo–Rudy I cell model and also the monodomain model coupled with the epicardial variant of the model of ten Tusscher et al. (2006). The comparison is done between the extended BACOLI software package and Chaste software package Pitt-Francis et al. (2009). For the Luo–Rudy I cell model, we obtained a speed-up of a factor of around three. For the epicardial variant of the model of ten Tusscher et al. (2006), we obtained a speed-up of a factor of around two.

5.2 Future Directions

From the results described in Chapter 4, two general future directions follow. Because of the robustness and the efficiency of the BACOLI software package, further extensions of this package can be considered as the next goal. These extensions are divided into two different directions.

As a first direction, one can consider extending the BACOLI software package such that it solves other types of equations. Presently, the extended BACOLI software package can only solve parabolic PDEs and PDEs with no dependency on the spatial derivatives. It can be extended further to solve systems of parabolic PDEs coupled with elliptic or hyperbolic PDEs. The *bidomain* model Tung (1978) are an example

that includes elliptic equations. This model is defined as follows:

$$\begin{aligned}
\chi C_m \frac{\partial v}{\partial t} + \chi I_{\text{ion}}(\boldsymbol{s}, v, t) &= \nabla \cdot (\sigma_i \nabla v) + \nabla \cdot (\sigma_i \nabla u_e)\,, \\
0 &= \nabla \cdot (\sigma_i \nabla v) + \nabla \cdot ((\sigma_i + \sigma_e)\, \nabla u_e)\,, \\
\frac{\partial \boldsymbol{s}}{\partial t} &= \boldsymbol{f}(\boldsymbol{s}, v, t),
\end{aligned}$$

with boundary conditions

$$\begin{aligned}
\hat{\boldsymbol{n}} \cdot (\sigma_i \nabla v + \sigma_i \nabla u_e) &= 0, \\
\hat{\boldsymbol{n}} \cdot (\sigma_e \nabla u_e) &= 0,
\end{aligned}$$

where v is the transmembrane potential, u_e is the extracellular potential, and $\boldsymbol{s}$ is a vector of cellular states such as gating variables and ionic concentrations. The function $\boldsymbol{f}(\boldsymbol{s}, v, t)$ is a non-linear function describing cellular dynamics, $I_{\text{ion}}(\boldsymbol{s}, v, t)$ is the ionic current per cell membrane area, and σ_i and σ_e are extracellular and intracellular conductivity tensors, respectively. The quantity χ is the area of the cell membrane per unit volume, and C_m is the capacitance of the cell membrane per unit area. Finally, $\hat{\boldsymbol{n}}$ is the unit outward normal. See, e.g., Sundnes et al. (2006), for further details on the bidomain model.

Because the extended `BACOLI` software package cannot solve systems with more than one PDE, one can extend it further to do so and solve multi-scale advection-reaction-diffusion equations such as a one-dimensional mathematical model of concrete-rewetting Chapwanya et al. (2009). This mathematical model describes the physical and chemical structure of dry concrete that is exposed to a hydration front Chapwanya et al. (2009). This process is done to make a stronger concrete Hall (2007). It consists of non-linear advection, non-linear diffusion, and non-linear reaction terms. Depending on how detailed is the model, there are variety of models that describe

this phenomenon. The equations for the rewetting model can be written as:

$$\begin{aligned}
\frac{\partial\theta}{\partial t} &= \frac{\partial}{\partial x}\left[D(\theta,\epsilon)\frac{\partial\theta}{\partial x}\right] - \nu(\theta-\theta_r)^+\frac{m_w r_{\text{csh}}}{\rho_w m_{\text{csh}}},\\
\frac{\partial(\theta C_\alpha)}{\partial t} &= \frac{\partial}{\partial x}\left(\theta D_\alpha\frac{\partial C_\alpha}{\partial x}\right) - \frac{\partial(uC_\alpha)}{\partial x} - (\theta-\theta_r)^+ r_\alpha,\\
\frac{\partial(\theta C_\beta)}{\partial t} &= \frac{\partial}{\partial x}\left(\theta D_\beta\frac{\partial C_\beta}{\partial x}\right) - \frac{\partial(uC_\beta)}{\partial x} - (\theta-\theta_r)^+ r_\beta,\\
\frac{\partial(\theta C_q)}{\partial t} &= \frac{\partial}{\partial x}\left(\theta D_q\frac{\partial C_q}{\partial x}\right) - \frac{\partial(uC_q)}{\partial x} + (\theta-\theta_r)^+(r_{\text{csh}} - k_{\text{prec}}C_q + k_{\text{diss}}C_g),\\
\frac{\partial(\theta C_g)}{\partial t} &= (\theta-\theta_r)^+(k_{\text{prec}}C_q - k_{\text{diss}}C_g),
\end{aligned}$$

where $(\theta-\theta_r)^+ = \max(0,\theta-\theta_r)$; θ is the volumetric water content; C_α, C_β, C_q, and C_g are the respective constituent concentrations of C_3S in concrete, C_2S in concrete, calcium-silicate hydrate (C-S-H) in liquid, and solid C-S-H gel; r_α, r_β, and r_{csh} are functions that govern the rates of reaction; D is the effective diffusivity; D_α, D_β, and D_q are the respective diffusivities of C_3S in concrete, C_2S in concrete, and C-S-H in liquid. The remaining variables are substance-specific constants, which are defined in Chapwanya et al. (2009).

A second direction is to modify the `BACOLI` software package such that more speed-up is achieved. Based on the principle that large algorithms can be broken into smaller ones, some parts of the `BACOLI` software package can be broken into smaller parts such that many calculations be carried out simultaneously. For example, the error estimate mentioned in (2.21) and (2.22) can be calculated simultaneously. Each of the components of these two error estimates can be calculated simultaneously as well. On the other hand, setting up and solving the ABD matrices presented in the Figure 2.12 can be done in parallel. In general case, in (2.15) there are N_x subintervals. Thus there are (N_x-1) interior points and accordingly there are (N_x-1) interior blocks in the ABD matrices. Because each interior block corresponds to an individual subinterval, the elements of these sub-blocks may be determined in parallel. In this fashion, along with the high-order method already applied in the `BACOLI` software package, more speed-up can be achieved. Some parallel solution techniques for ABD systems are proposed in Amodio and Romanazzi (2009); Amodio

et al. (1993). These techniques can be considered in the BACOLI software package to obtain further speed-up.

References

Alscher, C., and W. Beyn (1998), Simulating the motion of the leech: a biomechanical application of DAEs, *J. Numer. Algorithms*, *19*(1–4), 1–12.

Amodio, P., and G. Romanazzi (2009), Parallel numerical solution of ABD and BABD linear systems arising from BVPs., *SCPE*, *10*(4), 373–383.

Amodio, P., J. R. Cash, G. Roussos, R. W. Wright, G. Fairweather, I. Gladwell, G. L. Kraut, and M. Paprzycki (1993), Almost block diagonal linear systems: sequential and parallel solution techniques, and applications, *Numer. Linear Algebr.*, *1*(1), 1–7.

Arsenault, T., P. H. Muir, and T. Smith (2009), Superconvergent interpolants for efficient spatial error estimation in 1D PDE collocation solvers., *Can. Appl. Math. Q.*, *17*(3), 409–431.

Arsenault, T., S. Tristan, P. H. Muir, and P. Keast (2011), Efficient interpolation-based error estimation for 1d time-dependent PDE collocation codes., *Tech. rep.*, Saint Mary's University Department of Mathematics and Computing Science.

Arsenault, T., T. Smith, P. H. Muir, and J. Pew (2012), Asymptotically correct interpolation-based spatial error estimation for 1D PDE solvers., *Can. Appl. Math. Q.*, *20*(3), 307–328.

Ascher, U. M., and L. R. Petzold (1998), *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, Springer-Verlag, New York, USA.

Ascher, U. M., R. M. M. Mattheij, and R. D. Russell (1995), *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, SIAM, Philadelphia.

Bailey, R. (2012), Cardiac cycle, `http://surgery.about.com/od/beforesurgery/a/HeartBloodFlow.htm`.

Bauer, I., H. G. Bock, S. Körkel, and J. P. Schlöder (2000), Numerical methods for optimum experimental design in DAE systems, *J. Comput. Appl. Math.*, *120*(1-2), 1–15.

Berzins, M., P. J. Capon, and P. K. Jimack (1998), On spatial adaptivity and interpolation when using the method of lines, *Appl. Numer. Math.*, *26*, 117–133.

Bondarenko, V. E., G. P. Szigeti, G. C. L. Bett, S. J. Kim, and R. L. Rasmusson (2004), Computer model of action potential of mouse ventricular myocytes., *Am. J. Physiol.-Heart C.*, *287*(3), H1378–H1403.

Brenan, K. E., S. L. Campbell, and L. R. Petzold (1996), *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, Philadelphia.

Chapwanya, M., W. Liu, and J. M. Stockie (2009), A model for reactive porous transport during re-wetting of hardened concrete, *J. Eng. Math.*, *65*, 53–73.

Cherry, E. M., H. S. Greenside, and C. S. Henriquez (2000), A space-time adaptive method for simulating complex cardiac dynamics, *Phys. Rev. Lett.*, *84*(6), 1343–1346.

Clements, C. J. (1996), Nonlinear wave propagation in an anisotropic medium, Master's thesis, Dalhousie University.

de Boor, C. (1977), Package for calculating with B-splines., *SIAM J. Numer. Anal.*, *14*(3), 441–472.

de Boor, C. (1978), *A Practical Guide to Splines*, Springer-Verlag, New York.

de Boor, C., and S. D. Conte (1980), *Elementary Numerical Analysis An Algorithmic Approach*, McGraw-Hill Higher Education.

de Boor, C., and T. M. Inc. (1999), *Spline Toolbox User's Guide*, The MathWorks, Inc., Natick, MA.

Diaz, J. C., G. Fairweather, and P. Keast (1983a), FORTRAN packages for solving certain almost block diagonal linear systems by modified alternate row and column elimination., *ACM T. Math. Software*, *9*(3), 358–375.

Diaz, J. C., G. Fairweather, and P. Keast (1983b), Algorithm 603: COLROW and ARCECO: FORTRAN packages for solving certain almost block diagonal linear systems by modified alternate row and column elimination, *ACM T. Math. Software*, *9*(3), 376–380.

Diaz, J. C., G. Fairweather, and P. Keast (1988), Remark on "Algorithm 603: COLROW and ARCECO: FORTRAN packages for solving certain almost block diagonal linear systems by modified alternate row and column elimination"., *ACM T. Math. Software*, *14*(2), 196.

Dodgson, N. (2013), B-splines, `http://www.cl.cam.ac.uk/teaching/2000/AGraphHCI/SMEG/node4.html`.

FitzHugh, R. (1961), Impulses and physiological states in theoretical models of nerve membrane, *Biophys. J.*, *1*(6), 445–466.

Hall, C. (2007), Anomalous diffusion in unsaturated flow: fact or fiction?, *Cement Concrete Res.*, *37*(3), 378–385.

Holder, D., L. Huo, and C. F. Martin (2007), *The Control of Error in Numerical Methods*, Springer-Verlag, New York.

Institute, A. B. (2013), The CellML project, `http://www.cellml.org/`.

Keast, P., and P. H. Muir (1991), Algorithm 688: EPDCOL: A more efficient PDECOL code, *ACM T. Math. Software*, *17*(2), 153–166.

Li, Z., and P. H. Muir (2013), B-spline Gaussian collocation software for two-dimensional parabolic PDEs., *Adv. Appl. Math. Mech.*, *5*(4), 528–547.

LLC, E. E. (2014), Control volume - Fluid flow, `http://www.engineersedge.com/fluid_flow/control_volume.htm`.

Luo, C., and Y. Rudy (1991), A model of ventricular cardiac action potential, *Circ. Res.*, *68*(6), 1501–1526.

Mackenna, B. R., and R. Callander (1997), *Illustrated Physiology (6th ed.)*, Churchill Livingstone, Edinburgh, UK.

Marsh, M. E., S. T. Ziaratgahi, and R. J. Spiteri (2012), The secrets to the success of the Rush – Larsen method and its generalizations, *IEEE T. Bio-med. Eng.*, *59*(9), 2506–2515.

Moore, P. K. (1995), Comparison of adaptive methods for one dimensional parabolic systems., *Appl. Numer. Math.*, *16*(4), 471–488.

Moore, P. K. (2001), Interpolation error-based a posteriori error estimation for two-point boundary value problems and parabolic equations in one space dimension., *Numer. Math.*, *90*(1), 149–177.

Moore, P. K. (2014), Papers and software, `http://faculty.smu.edu/pmoore/papers.html`.

Muir, P. H. (2013), B-spline Gaussian collocation software for 1D parabolic PDEs., in *Recent advances in scientific computing and applications. Eighth international conference on scientific computing and applications, University of Nevada, Las Vegas, NV, USA, April 1–4, 2012. Proceedings*, pp. 267–276, Providence, RI: American Mathematical Society (AMS), doi:10.1090/conm/586/11630.

Nagumo, J., S. Arimoto, and S. Yoshizawa (1962), An active pulse transmission line simulating nerve axon, *P. IRE*, *50*(10), 2061–2070.

Paprzycki, M., and I. Gladwell (1991), Solving almost block diagonal systems on parallel computers, *Parallel Comput.*, *17*, 133–153.

Petzold, L. R. (1982), A description of DASSL: a differential/algebraic system solver., *Tech. rep.*, Sandia National Labs., Livermore, CA (USA).

Petzold, L. R., and P. Lötstedt (1986), Numerical solution of nonlinear differential equations with algebraic constraints. II. practical implications, *SIAM J. Sci. Stat. Comp.*, *7*(3), 720–733.

Pitt-Francis, J., et al. (2009), Chaste: A test-driven approach to software development for biological modelling, *Comput. Phys. Commun.*, *180*(12), 2452–2471.

Pormann, J. B. (1999), A modular simulation system for the bidomain equations., Ph.D. thesis, Duke University.

Quan, W., S. J. Evans, and H. M. Hastings (1998), Efficient integration of a realistic two-dimensional cardiac tissue model by domain decomposition, *IEEE T. Biomed. Eng.*, *45*(3), 372–385.

Riaza, R. (2008), *Differential-Algebraic Systems: Analytical Aspects and Circuit Applications*, World Scientific, Singapore.

Shampine, L. F., I. Gladwell, and S. Thompson (2003), *Solving ODEs with MATLAB*, Cambridge University Press, New York.

Sundnes, J., G. T. Lines, X. Cai, B. F. Nielsen, K. A. Mardal, and A. Tveito (2006), *Computing the Electrical Activity in the Heart*, Springer, Berlin.

ten Tusscher, K. H. W. J., and A. V. Panfilov (2006), Alternans and spiral breakup in a human ventricular tissue model, *Am. J. Physiol. Heart Circ. Physiol.*, *291*(3), 1088 – 1100.

Trangenstein, J. A., and C. Kim (2004), Operator splitting and adaptive mesh refinement for the Luo-Rudy I model., *J. Comput. Phys.*, *196*(2), 645–679.

Tung, L. (1978), A bi-domain model for describing ischemic myocardial dc potentials., Ph.D. thesis, Massachusetts Institute of Technology.

Vries, G. D. (2014), What is mathematical modelling?, `http://www.math.ualberta.ca/~devries/erc2001/slides.pdf`.

Wang, R., P. Keast, and P. H. Muir (2004a), A high-order global spatially adaptive collocation method for 1-D parabolic PDEs., *Appl. Numer. Math.*, *50*(2), 239–260.

Wang, R., P. Keast, and P. H. Muir (2004b), BACOL: B-spline Adaptive COLlocation software for 1-D parabolic PDEs, *ACM T. Math. Software*, *30*(4), 454–470.

Wang, R., P. Keast, and P. H. Muir (2008), Algorithm 874: BACOLR - spatial and temporal error control software for PDEs based on high-order adaptive collocation, *ACM T. Math. Software*, *34*(3), 15:1–15:28.

Whiteley, J. P. (2006), An efficient numerical technique for the solution of the monodomain and bidomain equations, *IEEE T. Bio-med. Eng.*, *53*(11), 2139–2147.

Whiteley, J. P. (2007), Physiology driven adaptivity for the numerical solution of the Bidomain equations., *Ann. Biomed. Eng.*, *35*(9), 1510–1520.

Appendix

Proof of B-spline properties

The two properties of the B-splines (2.2) and (2.3) put forth in Chapter 2 are proven in this appendix. The formula for generating the values of the B-spline basis functions at different knots is as follows.

$$B_{j,0}(x) = \begin{cases} 1, & \text{if } x_j \leq x < x_{j+1}, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

$$B_{j,M}(x) = \frac{x - x_j}{x_{j+M} - x_j} B_{j,(M-1)}(x) + \frac{x_{j+M+1} - x}{x_{j+M+1} - x_{j+1}} B_{j+1,M-1}(x). \tag{2}$$

Considering this formula, and the assumption that any $\frac{0}{0}$ is deemed to be zero, the proof of the first property, $B_{1,M}(x_1) = 1$, by induction is as follows. Let M be the maximum degree of the B-splines. For $M = 1$ the order of the B-splines is 2 and the corresponding open uniform knot sequence is $\{u_1, u_1, u_2\}$. In order to fit this sequence according to the notations of the aforementioned formula, let $x_1 = u_1$, $x_2 = u_1$ and $x_3 = u_2$. For calculating $B_{1,1}(x_1)$ both of the indices j and M are one. Thus by the aforementioned formula,

$$B_{1,1}(x) = \frac{x - x_1}{x_2 - x_1} B_{1,0}(x) + \frac{x_3 - x}{x_3 - x_2} B_{2,0}(x),$$

and at the point $x = x_1$ the first fraction is $\frac{0}{0}$ thus zero and the second fraction is one because $x_1 = x_2$, giving rise to

$$B_{1,1}(x_1) = B_{2,0}(x_1),$$

again applying (1) with j and M to be 2 and 0 respectively,

$$B_{2,0}(x) = \begin{cases} 1, & \text{if } x_2 \leq x < x_3, \\ 0, & \text{otherwise,} \end{cases}$$

thus because $x_1 = x_2$, at the point $x = x_1$, $B_{2,0}(x_1) = B_{2,0}(x_2) = 1$. As a whole,

$$B_{1,1}(x_1) = B_{2,0}(x_1) = 1.$$

In a general case, suppose that for $M = l$ with $l \geq 2$, (1) holds, i.e., $B_{1,l}(x_1) = 1$. If the property is proven to hold for $M = l + 1$, i.e., if $B_{1,l+1}(x_1) = 1$, then the proof is complete.

The knot sequence corresponding to $B_{1,l+1}$ is

$$\{\underbrace{u_1, u_1, \cdots, u_1}_{(l+2) \text{ times}}, u_2\}.$$

In order to fit this sequence according to the notations of (1) and (2), let $x_1 = u_1$, $x_2 = u_1, \cdots, x_{l+2} = u_1$ and $x_{l+3} = u_2$, then

$$B_{1,l+1}(x) = \frac{x - x_1}{x_{l+2} - x_1} B_{1,l}(x) + \frac{x_{l+3} - x}{x_{l+3} - x_2} B_{2,l}(x),$$

and at the point $x = x_1$ the first fraction is $\frac{0}{0}$ (and thus taken to be zero) and the second fraction is one because $x_1 = x_2$, giving rise to

$$B_{1,l+1}(x_1) = B_{2,l}(x_1),$$

again applying (2) with j and M to be 2 and l respectively, yields

$$B_{2,l}(x_1) = B_{3,l-1}(x_1),$$

and this process continues untill the degree reaches zero,

$$B_{2,l}(x_1) = B_{3,l-1}(x_1) = B_{4,l-2}(x_1) = \cdots = B_{l+2,0}(x_1).$$

On the other hand, according to (1),

$$B_{l+2,0}(x) = \begin{cases} 1, & \text{if } x_{l+2} \leq x < x_{l+3}, \\ 0, & \text{otherwise,} \end{cases}$$

and because $x_1 = x_{l+2}$, then $B_{l+2,0}(x_1) = B_{l+2,0}(x_{l+2}) = 1$. As a whole,

$$B_{1,l+1}(x_1) = B_{l+2,0}(x_1) = 1,$$

this completes the proof.

The proof of the second property is based on a spline property mentioned in de Boor (1978). For the first derivative of a spline,

$$D\left(\sum_{j=r}^{s}(\alpha_j B_{j,M+1})\right) = M \sum_{j=r}^{s+1}\left(\frac{\alpha_j - \alpha_{j-1}}{t_{j+M} - t_j} B_{j,M}\right),$$

where α_{r-1} and α_{s+1} are deemed to be zero. Because the knot sequence is assumed to be of open uniform type, and the order is $M + 1$, then the interval $[x_1, \cdots, x_{n+M+1}]$ is equivalent to the interval $[x_{M+1}, \cdots, x_{n+1}]$. Applying the aforementioned property on $[x_{M+1}, \cdots, x_{n+1}]$, with $r = 1, s = 2, \alpha_1 = 1$ and $\alpha_2 = 1$ gives rise to

$$D\left(\sum_{j=1}^{2}(B_{j,M+1})\right) = M \sum_{j=1}^{3}\left(\frac{\alpha_j - \alpha_{j-1}}{t_{j+M} - t_j} B_{j,M}\right).$$

At the point $x = x_1$,

$$B'_{1,M+1}(x_1) + B'_{2,M+1}(x_1) = M\left(\frac{\alpha_1 - \alpha_0}{t_{M+1} - t_1} B_{1,M}(x_1) + \frac{\alpha_2 - \alpha_1}{t_{M+2} - t_2} B_{2,M}(x_1) + \frac{\alpha_3 - \alpha_2}{t_{M+3} - t_3} B_{3,M}(x_1)\right) \tag{3}$$

Because $\alpha_2 = \alpha_1$, the second fraction on the right hand side is zero. Applying divided difference scheme, it is shortly proven that both of $B_{1,M}(x_1)$ and $B_{3,M}(x_1)$ vanishes.

For $B_{1,M}(x_1)$ the knot sequence is $\{x_1, x_2, \cdots, x_{M+2}\}$ where because of open uniformity they all are the same knot. In this case, $g_M(x_1, x_i) = (\max\{0, x_1 - x_i\})^{(M-1)}$. In the divided difference M tableau for this case, $g_{M-1}(x_1, x_i) = 0,\ i = 1, 2, \ldots, (M+2)$, and therefore all of the divided differences from 1 to $(M + 1)$ are zero. Thus $M_1(x)$ that is defined as the divided difference k of the function $g_M(x, x_i)$ is zero at $x = x_1$. Generally, because $B_{i,M}(x) = (x_{i+M} - x_i)(-1)^M \mu_i(x)$ and because in this case $M_1(x_1)$ is zero, then $B_{1,M}(x_1)$ vanishes. For $B_{3,M}$ the similar strategy can be applied, the knot sequence for this B-spline basis function is $\{x_3, x_4, \cdots, x_{M+4}\}$ and $g_M(x_3, x_i) = (\max\{0, x_3 - x_i\})^{(M-1)}$. In the divided difference M tableau for this case, $g_{M-1}(x_3, x_i) = 0,\ i = 3, 4, \ldots, (M + 4)$, and therefore all of the divided differences from 1 to $(M+1)$ are zero. Accordingly in the equation (3) the right hand side is zero and $B'_{1,M+1}(x_1)+B'_{2,M+1}(x_1) = 0$, or in other words, $B'_{1,M+1}(x_1) = -B'_{2,M+1}(x_1)$. The same proof can be done for degree M, thus $B'_{1,M}(x_1) = -B'_{2,M}(x_1)$, and this completes the proof.

Note that because for (2.3) to hold, only the openness property suffices, the uniformity property of the knot sequence is not used in the proof.